VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ZÍSKÁVÁNÍ ZNALOSTÍ Z OBCHODNÍCH PROCESŮ

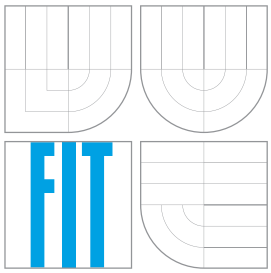
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAN SKÁCEL

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ZÍSKÁVÁNÍ ZNALOSTÍ Z OBCHODNÍCH PROCESŮ

BUSINESS PROCESS MINING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAN SKÁCEL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VLADIMÍR BARTÍK, Ph.D.

BRNO 2015

Abstrakt

Tato diplomová práce objasňuje disciplínu získávání znalostí z obchodních procesů. Je zde ukázán princip dolování. Značná část tématiky je věnována problémům při objevování procesu. Dále se práce věnuje analýze konkrétního výrobního procesu, na jejímž základě jsou navrženy tři dolovací metody, které se snaží zjistit nedostatky v procesu. První objevuje výrobní proces a vykresluje jej do grafu. Druhá metoda využívá simulátor produkční historie k získání produktů, které pravděpodobně způsobily opoždění výrobního procesu. Z takto získaných dat se poté dolují frekventované množiny. Třetí metoda se snaží predikovat čas zpracování produktu na vybraném pracovišti pomocí asociačních pravidel. Poslední dvě zmíněné metody využívají algoritmus Frequent Pattern Growth. Získané znalosti z této práce umožňují zefektivnit výrobní proces a lépe plánovat samotnou výrobu.

Abstract

This thesis explains business process mining and it's principles. A substantial part is devoted to the problems of process discovery. Further, based on the analysis of specific manufacturing process are proposed three methods that are trying to identify shortcomings in the process. First discovers the manufacturing process and renders it into a graph. The second method uses simulator of production history to obtain products that may caused delays in the process. Acquired data are used to mine frequent itemsets. The third method tries to predict processing time on the selected workplace using asociation rules. Last two mentioned methods employ an algorithm Frequent Pattern Growth. The knowledge obtained from this thesis improve efficiency of the manufacturing process and enables better production planning.

Klíčová slova

Obchodní proces, dolování procesů, Frequent Pattern Growth, frekventované množiny, simulace produkční historie, klasifikace, asociační pravidla.

Keywords

Business process, process mining, Frequent Pattern Growth, frequent itemsets, simulation of production history, clasification, association rules.

Citace

Jan Skácel: Získávání znalostí z obchodních procesů, diplomová práce, Brno, FIT VUT v Brně, 2015

Získávání znalostí z obchodních procesů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, PhD. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Skácel
25. května 2015

Poděkování

Chtěl bych poděkovat mému vedoucímu Ing. Vladimíru Bartíkovi, PhD. za odbornou pomoc při vypracování této práce. Dále bych chtěl poděkovat za udržení morálky mému spolubydlícímu Bc. Dušanovi Čtvrtníčkovi, bez kterého by tato práce nebyla dokončena.

Obsah

Kapitola 1

Úvod

Získávání znalostí z databází poskytuje nové prostředky k vylepšení procesů různých aplikačních domén. Existují dva hlavní předpoklady pro používaní této technologie. Prvním je dostatek informací o historii procesu, jelikož čím dál více událostí v systémech je zaznamenáváno. Avšak většina organizací diagnostikuje problémy pomocí fiktivních, ale ne reálných dat, která jsou do systémů ukládána. Druhým předpokladem je, že prodejci Business Inteligence aplikací a aplikací pro řízení procesu slibovali zázraky. Navzdory tomu, že tyto aplikace získaly mnoho pozornosti, tak nenaplnili očekávání akademiků, konzultantů a prodejců softwaru.

Získávání znalostí z procesů je nově vznikající disciplína, která poskytuje ucelenou sadu nástrojů pro získání znalostí na základě faktů a k podpoře vylepšování procesu. Tato disciplína své základy staví na modelech procesů a na obecném získávání znalostí. Avšak dolování procesů je více než jen sloučením existujících přístupů. Techniky pro získávání znalostí jsou příliš soustředěny na data, a tak neposkytují ucelený pohled na proces uvnitř organizace. Dolování procesů poskytuje různé dolovací techniky, které pomohou organizacím objevit nebo vylepšit jejich skutečný obchodní proces. Nejedná se pouze o objevení procesu, ale je možné testovat navržený proces s reálnou podobou nebo zjišťovat odchylky, predikovat spoždění, či poskytnout podporu pro důležitá rozhodnutí.

V této práci jsou získávány znalosti z výrobního procesu. Cílem je navrhnout a implementovat dolovací metody, které vylepší nebo urychlí výrobní proces. Získané informace musí být pochopitelné, dříve neznámé a potenciálně užitečné. V práci jsou navrženy dvě metody, které tyto požadavky splňují. Obě metody jsou založeny na frekventovaných množinách, které jsou získány pomocí algoritmu *Frequent Pattern Growth*. První metoda využívá simulátor produkční historie k získání produktů, které pravděpodobně způsobily opoždění výrobního procesu. Z takto získaných dat se poté dolují frekventované množiny. Druhá metoda se snaží predikovat čas zpracování produktu na vybraném pracovišti v procesu. K tomu se využívají asociační pravidla.

Struktura práce je následující. Kapitola 2 uvádí čtenáře do problematiky získávání znalostí z databází a představuje základní metody pro dolování z dat. Následující Kapitola 3 podrobněji popisuje získávání znalostí z obchodních procesů. Je zde ukázán princip techniky objevení procesu, která se považuje za jednu z nejtěžších a také budou představeny i některé možné problémy při objevování. V kapitole 4 budou vysvětleny dolovací algoritmy, které se používají v implementační části práce. Kapitola 5 se věnuje analýze procesu a na jejím základě jsou představeny navržené metody pro získání znalostí. Implementace výsledné aplikace bude ukázána v Kapitole 6. Poslední Kapitola 7 se zabývá výsledky dolování a experimenty s implementovanými metodami.

Kapitola 2

Získávání znalostí z databází

Získávání znalostí z databází (Knowledge Discovery in Databases) je proces extrakce zajímavých znalostí z velkého množství dat, přičemž jako zajímavá je zde chápána znalost, která je netriviální, skrytá, dříve neznámá a potenciálně užitečná. Získané znalosti se nejčastěji využívají k podpoře rozhodování. V současné době existuje velké množství oblastí, ve kterých se získávání znalostí z databází výrazně uplatňuje.

Příkladem může být zjištění informace o tom, jaké produkty si zákazníci kupují dohromady. Prodejce díky této informaci může vhodněji rozmístit zboží v prodejně nebo lépe cílit reklamu. Dalšími příklady využití procesu získávání znalostí z databází jsou finanční analýza, detekce podvodů, bioinformatika, web, zdravotnictví a mnoho dalších. Velmi často používaným alternativním termínem pro získávání znalostí je dolování z dat (data mining). Ve skutečnosti je však dolování z dat pouze jedním krokem procesu získávání znalostí.

Proces získávání znalostí z databází se skládá z několika fází. Tyto fáze jsou popsány v následující podkapitole. Podkapitola 2.2 popisuje nejběžnější zdroje dat pro získávání informací a podkapitola 2.3 představuje typy dolovacích úloh, které se využívají k extrakci znalostí. V této kapitole je čerpáno především z knih [13], [9] a z webu [12].

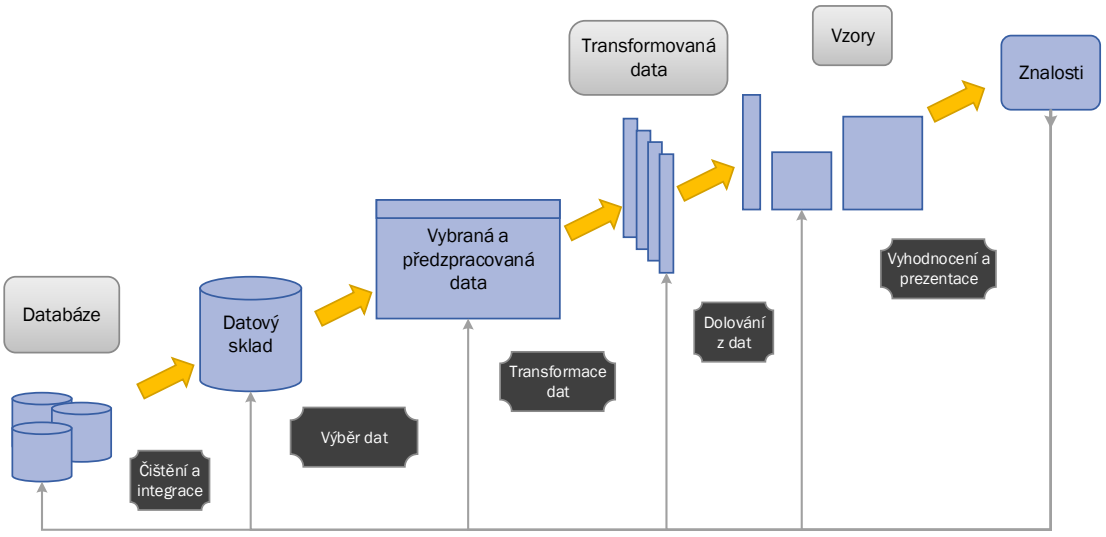
Historie získávání znalostí

Historie dolování dat je úzce spjata s historií databázových systémů. Je to dáno skutečností, že výhradním zdrojem dat pro dolování jsou právě databázové systémy. Počátky databázových systémů se datují do 60. let 20. století. V této době se používaly výhradně síťové a hierarchické databázové systémy. Tyto modely byly v 70. letech nahrazeny relačním modelem dat, jehož implementací je systém řízení báze dat. Tento systém byl v následujících letech dále vyvíjen a zdokonalován. Vznikly pokročilé databázové modely jako např. deduktivní databáze. V 90. letech již byly tyto systémy natolik vyvinuty, že vznikal tlak pro podporu pokročilejších operací (operace pro podporu rozhodování, získávání znalostí, analýzu dat atp.). V tomto období vznikají technologie datových skladů, algoritmy pro dolování z dat. Počátkem 21. století došlo k vývoji dolovacích algoritmů i nad jinými typy dat, jako jsou např. proudy dat.

2.1 Proces získávání znalostí

Získávání znalostí z databází je komplexní proces, který se skládá z řady iterativních částí. Během tohoto procesu se vstupní data zpracovávají, transformují a pomocí různých dolovacích algoritmů se z nich získávají užitečné informace. Pro dosažení přesnějších výsledků se

některé fáze procesu opakují vícekrát. Obrázek 2.1 zobrazuje celý proces získávání znalostí z databází.



Obrázek 2.1: Schéma procesu *získávání znalostí z databází*, převzato a upraveno z webu [12]

Popis jednotlivých fází procesu:

1. **Čištění dat** – snaha o doplnění chybějících hodnot, odstranění šumu v datech, identifikaci odlehlých hodnot a vyřešení nekonzistence dat.

2. **Integrace dat** – sloučení dat z různých datových zdrojů. Právě integrace dat je hlavním zdrojem nekonzistence, a proto se často integrace a čištění dat provádí v jednom kroku.

3. **Výběr dat** – cílem je vybrat data, která jsou relevantní pro danou dolovací úlohu. Data bývají často uložena v relačních databázích, proto výběr dat představuje selekci relevantních sloupců z tabulek. V případě datových skladů můžeme vybrat vhodné dimenze.

4. **Transformace dat** – úprava dat do podoby vhodné pro dolování. Může zahrnovat odstranění šumu, agregaci, generalizaci, normalizaci a konstrukci nových atributů. V případě potřeby může být provedena i redukce dat.

5. **Dolování z dat** – jádro procesu, kde je aplikován konkrétní algoritmus za účelem extrakce požadovaných vzorů. Algoritmy pro dolování vycházejí z různých disciplín jako je statistika, strojové učení, databázové technologie a jiné.

6. **Vyhodnocení vzorů** – cílem tohoto kroku je určit skutečně zajímavé vzory pomocí měr užitečnosti. Mezi tyto míry například patří spolehlivost a podpora.

7. **Prezentace znalostí** – výsledné informace se uživateli prezentují ve vhodné formě. Nejčastěji se využívají různé grafy a tabulky.

První čtyři kroky procesu jsou souhrnně označovány jako předzpracování dat. V případě datových skladů se také můžeme setkat s pojmenováním ETL - extrakce, transformace a přenos (loading). V této fázi se data připravují pro dolování.

2.2 Data pro dolování

Dolovat je v podstatě možné z datových zdrojů libovolného typu – ať už se jedná o data perzistentně uložená anebo dynamicky se měnící (tzv. datové proudy). Mezi typické zdroje dat pro dolování patří:

- **Relační databáze** – data uchovávaná v tabulkách. Řádky tabulek reprezentují jednotlivé záznamy a sloupce obsahují informace o atributech těchto záznamů. Pro práci s daty se využívá jazyk SQL.
- **Transakční databáze** – data jsou v souboru, ve kterém každý záznam reprezentuje jednu transakci. Transakce zahrnuje unikátní identifikátor a seznam položek, které transakci tvoří. Příkladem může být nákup zboží zákazníkem.
- **Datový sklad** – komplexní integrované datové úložiště, kde jsou data uložena ve struktuře, která umožňuje efektivní analýzu a dotazování. Práce s tímto úložištěm pak probíhá pomocí OLAP[1] operací.
- **Prostorové a časoprostorové databáze** – uchovávají informace vztahující se k nějakému prostoru. Typickým příkladem je geografická databáze obsahující souřadnice objektů na mapě.
- **Databáze sekvencí** – obsahuje uspořádané událostí, kde čas může, ale nemusí, být explicitně zaznamenám. Typickým příkladem jsou sekvence nákupů zboží.
- **Multimediální databáze** – uchovávají textové dokumenty, obrázky, audio a video.
- **Web** – velmi rozsáhlá heterogenní databáze. Typickými dolovacími úlohami nad tímto úložištěm je dolování z užití webu, automatické shlukování, klasifikace webových stránek a analýza komunit v sociálních sítích.

2.3 Typy dolovacích úloh

Podle typu výsledků můžeme dolovací úlohy rozdělit do dvou kategorií:

- **Deskriptivní** – výsledky nějákým způsobem popisují zdrojová data (vzory, shluky apod.).
- **Prediktivní** – výsledkem úlohy je model aplikovatelný na neznámá data s cílem predikovat určité vlastnosti.

Mezi základní dolovací úlohy patří popis konceptu nebo třídy, dolování frekventovaných vzorů a asociačních pravidel, klasifikace a predikce, shluková analýza, analýza odlehlých hodnot a evoluční analýza. O těchto úlohách budou stručně pojednávat následující část kapitoly.

[1] Online Analytical Processing

Popis konceptu nebo třídy

Cílem této dolovací úlohy je asociovat data s určitým konceptem nebo třídou. Pro uživatele pak může být užitečné jednotlivé třídy a koncepty popsat nějakým souhrnným, stručným a přitom přesným způsobem. Popis tříd a konceptů lze získat pomocí následujících metod:

- **Charakterizace dat** – třída je popsána sumarizací jejích obecných vlastností.
- **Diskriminace dat** – třída je popsána na základě srovnání s jinou třídou nebo množinou tříd.

Dolování frekventovaných vzorů a asociačních pravidel

Frekventované vzory jsou vzory, které se v datech vyskytují často. Frekventované vzory se objevují v mnoha různých podobách, například jako frekventované množiny u transakčních databází, (frekventované) sekvenční vzory, frekventované podgrafy a další. Dolování frekventovaných vzorů odhaluje zajímavé asociace a korelace[2] v datech. Klasickým příkladem dolování asociačních pravidel je analýza nákupního košíku, kdy jsou vyhledávány asociace mezi koupenými produkty z transakčních databází v obchodním řetězci.

V rámci dolování frekventovaných vzorů definujeme několik základních pojmů:

Definice 1 *Nechť $I = i_1, i_2, ..., i_n$ je množina položek. Nechť D je databáze transakcí, kde každá transakce T je množina položek taková, že $T \subseteq I$. Každé transakci přísluší unikátní identifikátor. Asociační pravidlo je implikace ve tvaru $A \Rightarrow B$, kde $A \subset I, B \subset I$ a $A \cap B = \emptyset$. Podpora (support) a spolehlivost (confidence) pravidla $A \Rightarrow B$ v množiňe transakcí D je pak s využitím pravděpodobnosti definována následovně:*

$$podpora(A \Rightarrow B) = P(A \cup B)$$

$$spolehlivost(A \Rightarrow B) = P(B \mid A) \tag{2.1}$$

Z Rovnice 2.1 lze dále odvodit následující rovnici pro výpočet spolehlivosti:

$$spolehlivost(A \Rightarrow B) = P(B \mid A) = \frac{podpora(A \cup B)}{podpora(A)} \tag{2.2}$$

Jinými slovy pravidlo $A \Rightarrow B$ má podporu p, pokud se v p % transakcí z D objevuje množina položek $A \cup B$. Totéž pravidlo má spolehlivost s, pokud s % transakcí z D, které obsahují A, obsahují také B. K odlišení zajímavých pravidel a korelací jsou při dolovacích úlohách definovány hodnoty pro minimální podporu a minimální spolehlivost, kterých musí získané pravidlo dosahovat, aby bylo považováno za zajímavé.

Dolování asociačních pravidel je pak tvořeno dvěma kroky:

1. Nalezení všech frekventovaných množin, které splňují podmínku minimální podpory. Pro realizaci tohoto kroku je možné použít například algoritmus Apriori či jeho modifikace. Tento algoritmus využívá přístupu generování a testování kandidátů na frekventované množiny s využitím tzv. Apriori vlastnosti, která říká, že každá neprázdná

[2] Korelace znamená vzájemný vztah mezi dvěma procesy nebo veličinami.

podmnožina frekventované množiny musí být také frekventovaná. Hlavními nedostatky tohoto algoritmu je nutnost mnoha průchodů databáze a příliš velký počet generovaných kandidátů. Druhý nedostatek odstraňuje například algoritmus Frequent Pattern Growth (FP Growth), který se generování kandidátů vyhýbá. FP Growth bude popsan dále v Kapitole 4.2.

2. Generování asociačních pravidel z frekventovaných množin tak, aby získaná pravidla splňovala podmínku minimální podpory a minimální spolehlivosti. Pro realizaci tohoto kroku se nejčastěji využívá Rovnice 2.2 pro výpočet spolehlivosti ze známých hodnot podpory.

Asociační pravidla se dále mohou klasifikovat podle nejrůznějších kritérií. Typy pravidel můžeme dělit podle:

- **Typu hodnot v pravidlech** – booleovská asociační pravidla (zajímá nás jen přítomnost, resp. nepřítomnost položky) a kvantitativní pravidla (popis asociace mezi kvantitativními položkami).
- **Počtu dimenzí v pravidlech** – jednodimenzionální (např. $koupi(A) \Rightarrow koupi(B)$) a vícedimenzionální (např. $prijem(10000-20000)+vek(30-39) \Rightarrow koupi(Automobil)$) asociační pravidla.
- **Úrovně abstrakce** – pokud pravidla obsahují různé úrovně abstrakce, tzv. víceúrovňová asociační pravidla.
- **Další rozšíření** – speciální rozšíření, např. maximální vzory, uzavřené množiny,

Klasifikace a predikce

Klasifikace je proces hledání modelu, který rozděluje data na základě jejich vlastností do konečné množiny tříd nebo konceptů. Klasifikace se používá k předpovědi hodnot kategorického charakteru. Predikce je obdobný proces jako klasifikace, jen s tím rozdílem, že se používá pro dedukci hodnot spojitého charakteru.

Celý proces klasifikace a predikce se skládá z následujících kroků:

- **Trénování** – v této fázi se vytvoří klasifikační model. Model je vytvořen na základě trénovací množiny dat. U dat z této množiny známe třídu, do které patří.
- **Testování** – v druhé fázi se otestuje přesnost klasifikačního modelu na datech z testovací množiny dat. U dat z této množiny také známé třídu, do které patří. Podle výsledku testování klasifikátoru se rozhodne, jestli je model dostatečně přesný a může být použit pro klasifikaci budoucích hodnot.
- **Aplikace** – v poslední fázi se získaný model využívá pro klasifikaci objektů, jejichž cílové třídy neznáme.

Mezi základní metody používané pro klasifikaci a predikci patří například:

- **Rozhodovací strom** – jedná se o graf stromové struktury, kde každý vnitřní uzel reprezentuje test hodnoty některého atributu, každá větev reprezentuje výsledek testu a každý listový uzel reprezentuje třídu, do níž je daný objekt klasifikován.

- **Bayesovská klasifikace** –jedná se o klasifikaci založenou na statistice, konkrétně na výpočtu pravděpodobnosti příslušnosti objektu do jednotlivých tříd.
- **Neuronová síť** – pro klasifikaci se využívá nejčastěji neuronové sítě se zpětným šířením chyby. Jedná se o síť umělých neuronů, přičemž učení zde probíhá postupnou modifikací vah vstupů jednotlivých neuronů.
- **SVM (Support Vector Machines)** – slouží pro klasifikaci lineárních i nelineárních dat. Algoritmus je založen na nelineárním mapování trénovací množiny do vícedimenzionálního prostoru, v němž hledá optimální lineární oddělující rovinu, která maximalizuje vzdálenost mezi jednotlivými třídami.
- **Klasifikace založená na pravidlech** - model je reprezentován jako množina pravidel tvaru IF podmínka THEN třída, která jsou obvykle extrahována z rozhodnovacího stromu.
- **Klasifikace založená na k-nejbližším sousedství** – klasifikovaný vzorek je přiřazen do třídy, která je nejčetnější v množině k vzorků vybraných z trénovací množiny, které jsou klasifikovanému vzorku nejblíže vzhledem k určité vzdálenostní metrice.
- **Regrese** – jedná se o metodu nejčastěji využívanou pro predikci. Základní variantou je jednoduchá lineární regrese, která je založena na metodě nejmenších čtverců. Zobecněním této metody je násobná lineární regrese. Dalším typem pak je nelineární regrese, kterou však lze většinou transformovat na regresi lineární.

Dolování shluků

Podobně jako u klasifikace jsou i při shlukové analýze data rozřazována do tříd, ovšem v případě shlukování neznáme předem cílové třídy a u některých algoritmů neznáme ani jejich počet. Cílem shlukování je tedy rozdělit data do shluků/tříd na základě jejich podobnosti. Obsahem jednoho shluku jsou objekty navzájem podobné a zároveň rozdílné od objektů v ostatních shlucích. Pro určování podobnosti se používá nejčastěji tzv. vzdálenostní funkce (např. klasická Euklidovská vzdálenost).

Metod pro shlukování je opět celá řada:

- **Metody založené na rozdělování** – tyto metody rozdělují n objektů do k shluků, přičemž každý shluk musí obsahovat alespoň jeden objekt a každý objekt musí patřit pouze do jednoho shluku, (k-means, k-menoids).
- **Hierarchické metody** – metody tohoto typu vytvářejí hierarchický rozklad množiny objektů. Shlukování může probíhat přístupem zdola-nahoru (shlukující metoda), kdy na počátku každý objekt patří do vlastní třídy a postupně dochází ke slučování nejpodobnějších shluků. Opakem je přístup shora-dolů (rozdělující metoda), kdy na počátku patří všechny objekty do jednoho shluku, který je postupně dělen na menší shluky.
- **Metody založené na hustotě** – tyto metody vycházejí z předpokladu, že shluky jsou tvořeny oblastmi s vysokou hustotou objektů v prostoru dat, které jsou navzájem odděleny oblastmi s nízkou hustotou objektů (DBSCAN, DENCLUE).
- **Metody založené na mřížce** – tyto metody rozdělují prostor dat na konečný počet buněk, které tvoří mřížkovou strukturu. Hlavní výhodou je rychlá doba zpracování.

- **Metody založené na modelech** – cílem je nalézt co nejlepší shodu mezi datovou množinou a daným matematickým modelem.

Analýza odlehlých hodnot

Databáze může obsahovat datové objekty, které se neshodují s obecným chováním nebo modelem dat. Tyto data se nazývají odlehlé objekty. Většina dolovací úloh odlehlé objekty zahazuje jako datový šum nebo výjimky. Některé úlohy, jako je například detekce neobvyklého chování a podvodů, naopak považují odlehlé hodnoty za zajímavé a využívají analýzu odlehlých hodnot ve fázi dolování. Odlehlé hodnoty se získávají zpravidla použitím upravených dolovacích metod. Příkladem může být shlukování, které tyto objekty umí identifikovat. Další možností je použití metod statistického zpracování dat.

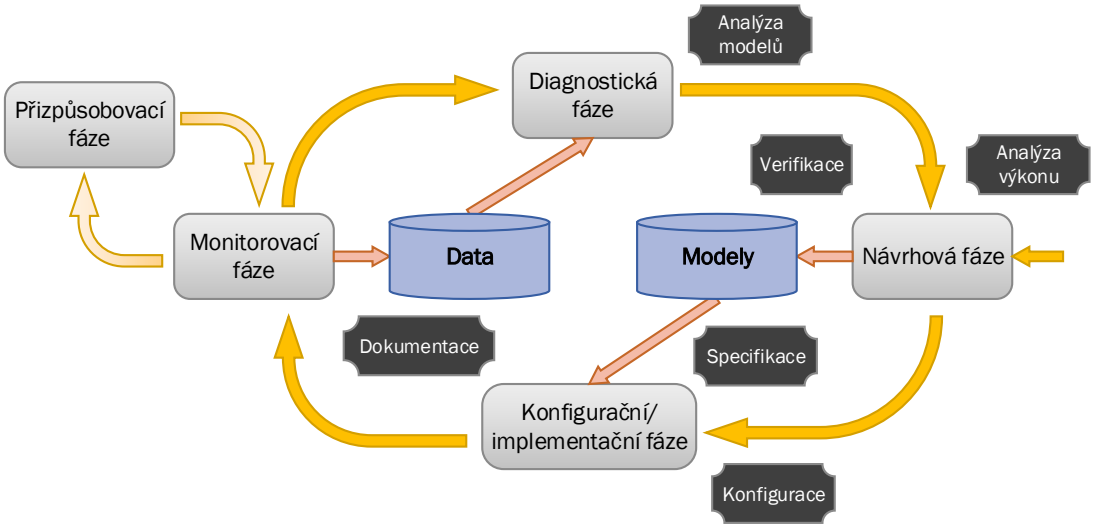
Evoluční analýza

Evoluční analýza popisuje a modeluje pravidelnosti nebo trendy objektů, jejichž chování se mění v čase. Mezi typické úlohy tohoto typu patří například analýza trendu či podobnostní hledání v časových řadách a dolovaní sekvenčních a periodických vzorů.

Kapitola 3

Získávání znalostí z obchodních procesů

Než přistoupíme k popisu dolování procesů, tak si nejprve objasníme životní cyklus procesního řízení (Business Process Management life-cycle) podle Obrázku 3.1. Životní cyklus popisuje odlišné řídící fáze konkrétních obchodních procesů. Informace v této kapitole jsou převzaty z [1], [2], [4], [3] a [15] .



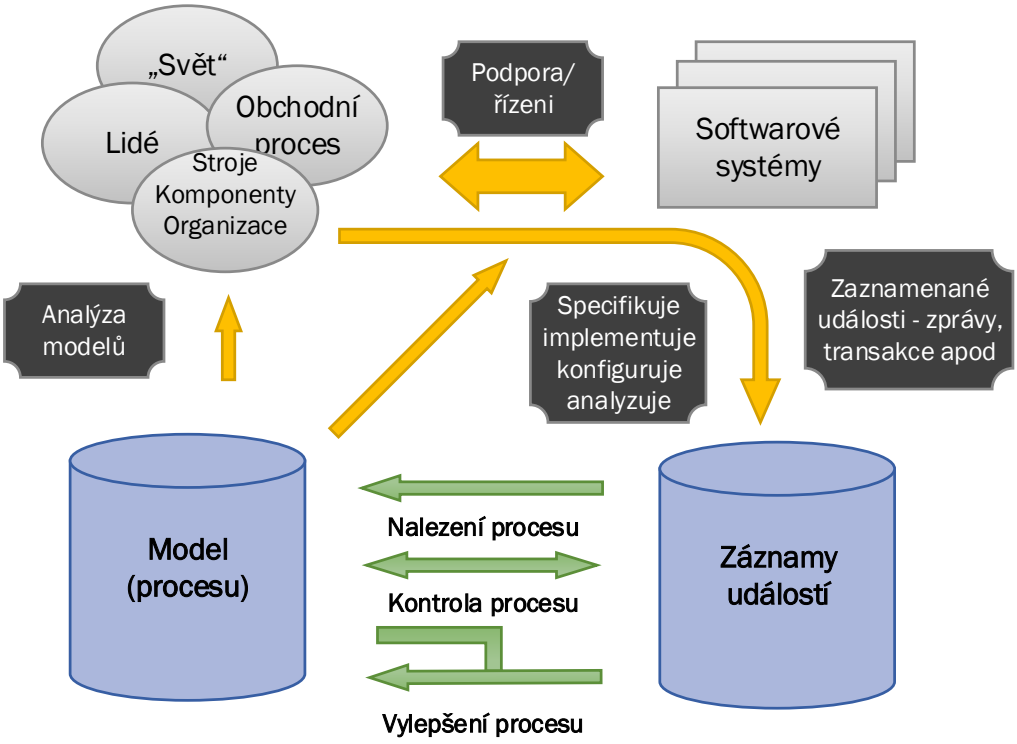
Obrázek 3.1: Životní cyklus řízení procesu, převzato a upraveno z knihy [1].

- **Návrhová fáze** – v této fázi se proces navrhnuje.
- **Implementační/konfigurační fáze** – model je převeden na běžící systém. Tato fáze může trvat velice krátkou dobu, pokud je model již ve spustitelné verzi a procesně řídící systém funguje správně. Avšak pokud je model pouze neformální a musí být naimplementován do existujicího softwaru, může tato fáze trvat značný čas.
- **Monitorovcí fáze** – proces normálně běží a je monitorován, aby se zjistilo, jestli není potřeba nějaká změna.

- **Přizpůsobovácí fáze** – některé změny jsou vyřešeny právě v této fázi. Celý proces není změněn a žádný nový software není přidán. Proces se pouze lehce upravuje a adaptuje pomocí předdefinovaných úkonů.
- **Diagnostická fáze**– v této fázi se vyhodnocuje proces a monitorují se vznikající požadavky způsobené změnou prostředí procesu (změny zákonů, práv, odpovědností). Nízký výkon nebo nové požadavky na proces mohou spustit novou iteraci životního cyklu, která startuje v návrhové fázi.

Jak je vidět z Obrázku 3.1, modely procesů hrají dominantí roli v návrhové fázi a v implementační/konfigurační fázi. Donedávna bylo velmi malé propojení mezi vyprodukovanými daty, které se do systému uložily během práce systému, a mezi skutečným navrhováním procesu. Ve většině organizací diagnostická fáze není podporována systematicky a opakovaně. Jen některé problémy a velké vnitřní změny způsobí nastartování nové iterace životního cyklu a informace z aktuálnho procesu nejsou použity v návrhové fázi. Dolování procesů poskytuje možnost opravdu uzavřít životní cyklus procesního řízení. Data zaznamenána v informačních systémech mohou být použita pro lepší náhled na aktuální proces, například se analyzují odchylky a tím se zvyšuje kvalita výsledného modelu.

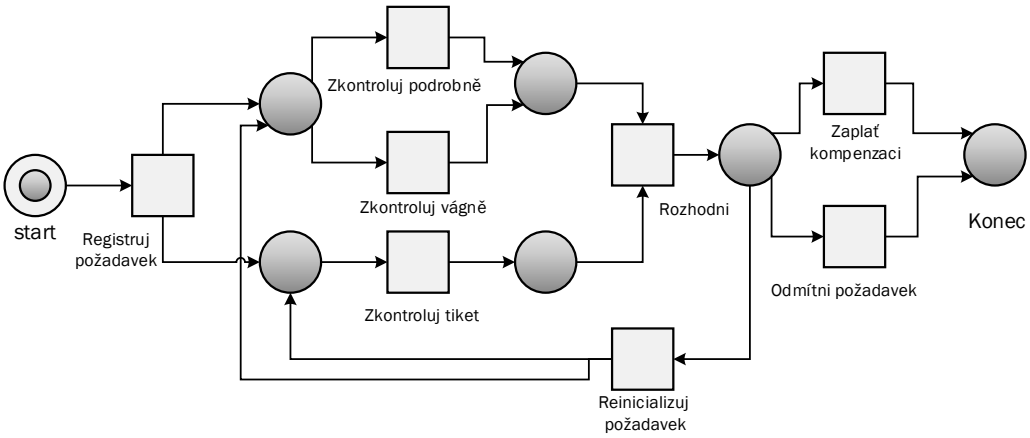
Dolování procesů je poměrně nová vědecká disciplína. Řadí se mezi strojové učení a získávání znalostí z databází na jedné straně a mezi modelování a analýzu procesů na straně druhé. Cílem dolování procesů je objevení, monitorování a vylepšení reálných procesů (ne smyšlených) pomocí získávání znalostí ze záznamů událostí, které jsou dostupné ve většině současných systémů.



Obrázek 3.2: Schéma *dolování procesů*

Obrázek 3.2 ukazuje, že dolování procesů tvoří spojení mezi skutečným procesem a jeho daty na jedné straně a mezi modelem procesu na straně druhé. Dnešní informační systémy ukládají obrovské množství událostí, které se podle Obrázku 3.2 nazývají záznam událostí. Avšak většina systémů ukládá tyto informace v nestrukturované formě, tzn. data o událostech jsou rozptýleny v mnoha tabulkách nebo v různých podsystémech. V takových případech data o událostech existují, ale je potřeba nějaké úsilí k jejich získání. Extrakce dat je interní část každého dolování procesů.

Předpokládejme, že je možné sekvenčně zaznamenávat události tak, že každá událost odkazuje na *aktivitu* (plně definovaný krok v procesu) a je spojena s konkrétní *instancí procesu* (jedna možná cesta procesem). Uvažujme například zpracování požadavků pro kompenzaci, které byly namodelovány pomocí Petriho sítě na Obrázku 3.3. Instance procesu jsou jednotlivé požadavky a pro každou instanci se ukládá cesta událostí. Příkladem uložené cesty může být posloupnost: *registruj požadavek, zkontroluj vágně, zkontroluj tiket, rozhodni, reinicializuj požadavek, zkontroluj tiket, zkontroluj důkladně, rozhodni, zaplať kompenzaci*. Zde jsou použita jména aktivit pro identifikaci událostí. Avšak, jsou zde dvě události *rozhodni*, které se staly v různém čase (čtvrtá a osmá událost), vyprodukovaly různé výsledky a mohly být provedeny různými lidmi. Zřejmě je důležité odlišit tyto dvě události. Kvůli tomu mnoho záznamů událostí navíc obsahuje další informace. Kdykoliv je to možné, techniky pro dolování procesů využívají extra informace jako jsou *zdroje* (lidé, zařízení) vykonávající danou aktivitu, *časové razítko* události, nebo *datové elementy* uložené spolu s událostí (velikost objednávky, atributy výrobku apod.).



Obrázek 3.3: Ukázka možné reprezentace procesu pomocí petriho sítě

3.1 Typy dolování procesu

Podle Obrázku 3.2 se dá záznam událostí použít pro tři různé typy dolování procesu. V této podkapitole jsou informace převzaty z knihy [1] a článku [2].

Nalezení procesu (discovery)

Technika pro nalezení procesu vezme záznam událostí a vytvoří model bez použití předem známých informací. Příkladem může být α - algoritmus. Tento algoritmus převede záznam

událostí na Petriho síť, která přesně modeluje chování zaznamenané v záznamu. Například pokud bychom jako vstupní parametr pro α - algoritmus dali dostatečný počet provedení procesu ukázaného na Obrázku 3.3, je tento algoritmus schopen automaticky sestrojit Petriho síť bez dalších znalostí. Pokud záznam událostí obsahuje navíc informace o zdrojích, je možné sestrojit také model zavísloti zdrojů, např. sociální síť, jak lidí v organizaci komunikují a pracují.

Kontrola procesu (conformance)

V této dolovací technice se vezme existující model procesu a porovná se se záznamem událostí toho stejného procesu. Této techniky se využíva pro zjišťení, zda realita (uložená v záznamu událostí) odpovída modelu. Lze také zkoumat zda model odpovída realitě. Například můžeme mít model procesu, ve kterém jsou potřeba dvě kontroly při platbě více jak jedním milionem korun. Analýzou záznamu událostí se zjistí, zda je toto pravidlo dodrženo nebo ne. Dalším příkladem může být kontrola tzv. pravidla čtyř očí, které říká, že jedna důležitá aktivita by neměla být vykonávána jednou a tou samou osobou. Pomocí porovnání záznamu událostí s modelem splňujícím dané požadavky můžeme nalézt potenciální případy podvodu. Tedy kontrola procesu může být použita pro detekci, lokalizaci a objasnění odchylek. Dále potom pro měření vážnosti těchto odchylek.

Vylepšení procesu (enhancement)

Cílem této techniky je rozšíření nebo vylepšení existujícího procesu pomocí informací ze záznamu událostí o aktuálním procesu. Rozlišují se dva typy vylepšení procesu:

- **Oprava** – modifikace modelu tak, aby lépe odpovídal realitě. Například mějme dvě aktivity, které jsou namodelovány sekvenčně. Avšak ve skutečnosti mohou nastat v jakémkoliv pořadí. Poté může být model opraven, aby lépe vyjadřoval realné chování.
- **Rozšíření** – přidání nové perspektivy do modelu procesu pomocí korelace se záznamem. Příkladem může být rozšíření modelu procesu o výkonostní data. Konkrétně přidáním časových razítek do záznamu událostí procesu *zpracování požadavku pro kompenzaci* z Obrázku 3.3 docílíme toho, že jsme schopni ukázat překážky, urovně služeb, časovou propustnost nebo frekvence opakování. Podobně může být zmíněný proces rozšířen o zdroje, rozhodovací pravidla, metriky kvality apod.

Model procesu z Obrázku 3.3 ukazuje pouze perspektivu *řídícího toku*. Avšak, pokud se rozšiřuje model procesu, přidávájí se nové perspektivy. Techniky *nalezení procesu* a *kontrola procesu* nejsou limitovány pouze na perspektivu *řídícího toku*. Například je možné nalézt sociální síť a zkontrolovat validitu této sítě s nějakým organizačním modelem pomocí záznamu událostí. Tedy k již zmíněným třem typům dolovacích technik můžeme přidat čtyři různé perspektivy dolování:

- **Perspektiva řídícího toku** – zaměřuje se na řídící tok, tedy na seřazení jednotlivých aktivit za sebou. Cílem této perspektivy je nalezení úplné charakteristiky všech možných cest. Bývá vyjádřena pomocí Petriho sítě, BPMN (Business Process Model and Notation), EPN (Event-driven Process Chain) nebo UML AD (Unified Modeling Language Activity diagram)

- **Organizační perspektiva** – zaměřuje se na informace o zdrojích, které jsou skryty v záznamu událostí. Zabývá se účastníky procesu, tedy lidmi, systémy, rolemi, odděleními apod. Zkoumá u nich, jak jsou zapojeni do procesu a jak spolu souvisí. Cílem je vytvořit strukturu organizace pomocí klasifikace lidí a jejich rolí nebo také představit sociální síť organizace.

- **Perspektiva instancí procesu** – pracuje s vlastnostmi jednotlivých instancí procesu, která je nejčastěji charakterizována cestou procesem. Avšak navíc instance procesu může být charakterizována pomocí hodnot odpovídajících datových elementů. Například pokud instance reprezentuje doplnění zboží do obchodu, může být velice zajímavé znát dodavatele jednotlivých produktů nebo počet objednaných kusů.

- **Časová perspektiva** – zabývá se časem a frekvencemi jednotlivých aktivit. Pokud se do záznamu událostí ukládají časové známky, je možné pomocí dolování odhalit překážky, měřit výkon systému, monitorovat využití zdrojů nebo predikovat zbývající čas běžících procesů.

Zmíněné perspektivy se částečně překrývají a nejsou úplné. Avšak poskytují dostatečnou charakteristiku aspektů, které chceme pomocí dolování procesů analyzovat.

V této práci bylo na příkladech ukázáno dolování procesů *offline*, tedy procesy byly analyzovány později, než se vykonaly. Nicméně, více a více technik pro dolování procesů lze použít i při *online* dolování. Těmto technikám se říka *podpora provozu* (operational support). Příkladem může být detekce neshody procesu s modelem při vzniku odchylky v systému za běhu tohoto systému. Dalším příkladem je predikce zbývajícího času běžící instance procesu. Máme-li částečně provedený proces, můžeme zbývající čas odhadnout na základě historických informací stejných nebo podobných instancí procesu. Tyto ukázky ilustrují fakt, že spektrum technik pro dolování procesů je široké a nejedná se pouze o techniku *nalezení procesu* (process discovery). V dnešní době může dolování procesů podporovat celý životní cyklus řízení procesu, jak je ukázáno na Obrázku 3.1. Dolování procesů není relevantní pouze pro fáze navrhování a diagnostiky, ale také pro monitorovací a přizpůsobovací fáze.

3.2 Ilustrace principu

K ilustraci principu dolování procesů použijeme příklad, který bude vycházet z Tabulky 3.4, která obsahuje záznam událostí. Příklad je převzat ze článku [3].

Identifikátor instance	Identifikátor úkolu
Instance 1	Úkol A
Instance 2	Úkol A
Instance 3	Úkol A
Instance 3	Úkol B
Instance 1	Úkol B
Instance 1	Úkol C
Instance 2	Úkol C
Instance 4	Úkol A
Instance 2	Úkol B
Instance 2	Úkol D
Instance 5	Úkol E
Instance 4	Úkol C
Instance 1	Úkol D
Instance 3	Úkol C
Instance 3	Úkol D
Instance 4	Úkol B
Instance 5	Úkol F
Instance 4	Úkol D

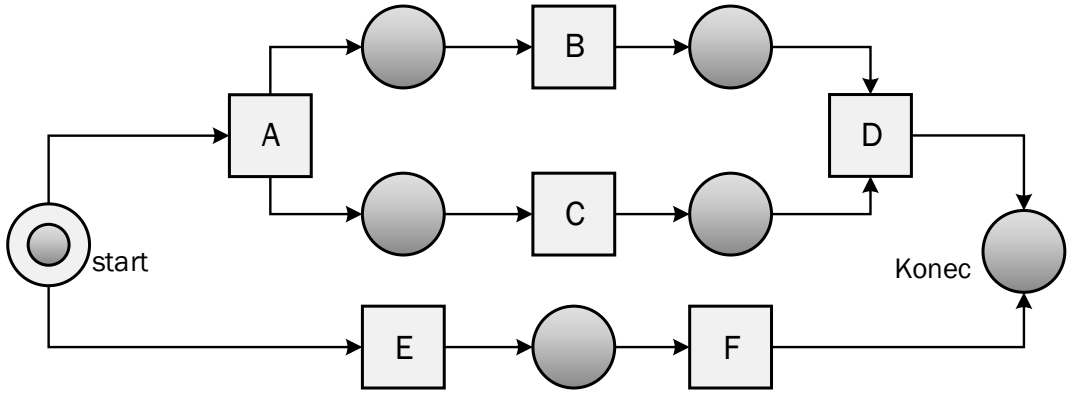
Obrázek 3.4: Příklad záznam událostí. Převzato z článku [3].

Tento záznam obsahuje informace o pěti instancích procesů. Záznam ukazuje, že u 4 instancí (1, 2, 3 a 4) byly provedeny úkoly A, B, C a D. U páté instance byly provedeny pouze dva úkoly, a to úkoly E a F. Ze záznamu dále můžeme vyčíst, že pokud je proveden úkol B, tak potom je vždy proveden i úkol C. Avšak v některých případech je úkol C proveden před úkolem B. Na základě informací z Tabulky 3.4 a po zvážení několika předpokladů o kompletnosti záznamu (například předpokládáme, že instance jsou reprezentativní, a že dostatečný počet instancí možného chování je zaznamenáno v tabulce), můžeme dedukovat model procesu, který je na Obrázku 3.5. Model je prezentován jako Petriho síť, kde jednotlivé úkoly jsou znázorněny přechody. Petriho síť začíná úkolem A a končí úkolem D. Po zpracování úkolu A se B a C zpracuje paralelně. V případě tohoto příkladu předpokládáme, že dva úkoly jsou paralelní, pokud se v záznamu vyskytují za sebou náhodně. Díky tomu, že rozlišujeme start a konec úkolu, je možné explicitně určit paralelizmus. Místo toho aby proces začínal úkolem A může začít i úkolem E následovaný F.

Tabulka 3.4 obsahuje minimum informací, které jsou potřeba pro úspěšné sestavení modelu. V mnoha aplikacích záznam procesů obsahuje časové razítko pro každý úkol a tato informace může být dále použita k zjištění dalších kauzálních [1] znalostí. Za další je možné

[1] Kauzalita je vztah mezi událostí (příčina) a druhou událostí (následek), kde první událost způsobuje druhou.

zkoumat relace mezi atributy jednotlivých instancí nebo skutečnou cestu, kterou každá instance prošla.



Obrázek 3.5: Model procesu korespondující se záznamem událostí

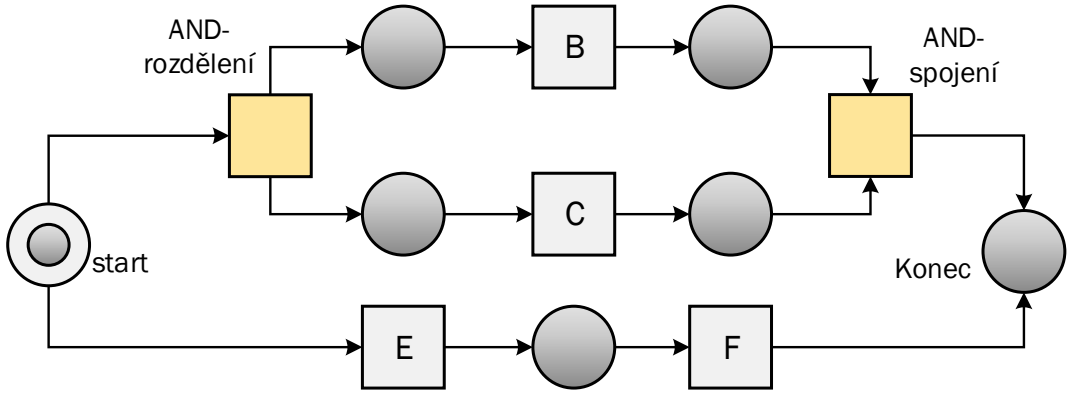
U tohoto příkladu je vcelku jednoduché vytvořit model procesu, který dokáže generovat záznam procesu. Avšak u větších modelů je to o dost obtížnější. Například, pokud by model projevoval známky alternativních a paralelních cest, poté záznam s největší pravděpodobností neobsahuje všechny možné kombinace. Vezměme si například 10 úkolů, které se mohou vykonat paralelně. Celkový počet možných cest sítí by byl $10! = 3628800$. Není možné předpokládat, že by všechny možné cesty byly zahrnuty v záznamu. A za další, některé cesty modelem mohou mít nižší pravděpodobnost vykonání a proto nemusí být nikdy detekovány. Zašumění dat může dále zkomplikovat vytváření modelu.

3.3 Možné problémy při dolování procesů

Dolování procesů přineslo řadu zajímavých vědeckých problémů. Některé již byly vyřešeny, jiné potřebují další výzkum. V této kapitole budou ukázány nejzajímavější problémy, které jsou více popsány v článku [4].

Dolování skrytých úkolů

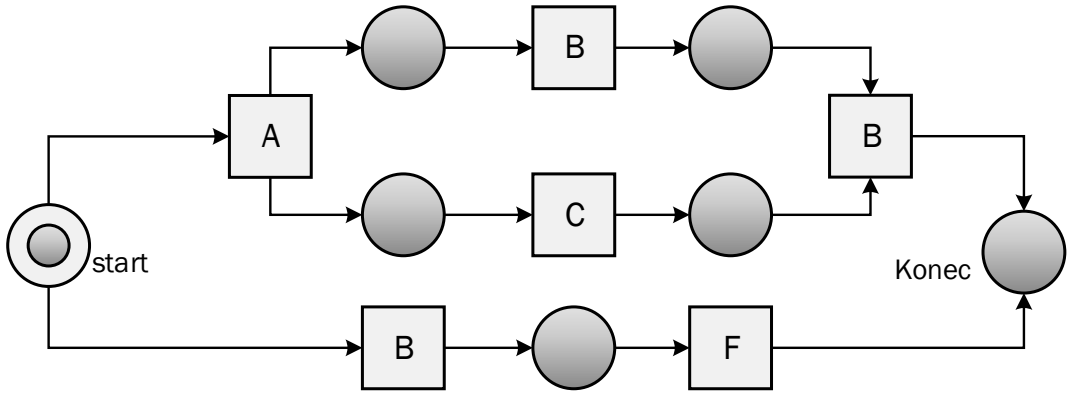
Jeden ze základních předpokladů dolování dat je, že všechny události jsou zaregistrovány v záznamu (výskyt úkolů pro specifické případy). Samozřejmě, není možné najít informaci o úkolech, které nejsou zaznamenány. Avšak vzhledem k specifickým jazykům je možné zaznamenávat tzv. skryté úkoly. Například předpokládejme, že v Tabulce 3.4 události, patřící k úkolu A, jsou odstraněny. Ačkoli záznam neodhalí úkol A, je zřejmé, že tam musí být *AND-rozdělení* pokud předpokládáme, že úkoly B a C jsou paralelní. Podobně můžeme předpokládat, že tam musí být *AND-spojení*, pokud bychom odstranili úkol D ze záznamu. Předpokládejme, že bychom odstranili úkoly A a D ze záznamu. V takovém případě je stále možné automaticky zrekonstruovat model procesu, Obrázek 3.6. Avšak pro komplikovanější procesy může být nalezení skrytých úkolů obtížný problém, který se řeší pozorováním nebo simulací.



Obrázek 3.6: Model procesu se dvěma skrytými úkoly.

Dolování duplikovaných úkolů

Problém duplikovaných úkolů souvisí se situací, kdy máme model procesu (např. Petriho síť) s dvěma uzly, které odpovídají jednomu úkolu. Předpokládejme, že v Tabulce 3.4 a na Obrázku 3.5 přejmenujeme úkol E na B. Výsledná Petriho síť je zobrazena na Obrázku 3.7. Můžeme vidět, že modifikovaný záznam by mohl být výsledkem modifikovaného procesu. Avšak je velice obtížné automaticky zkonstruovat model procesu z Tabulky 3.4, kde je přejmenováno E na B, protože není možné odlišit úkol B v instanci 5 od B v jiných instancích. Přítomnost duplikovaných úkolů je velice často spojena s přítomností skrytých úkolů. Mnoho procesů se skrytými úkoly ale bez duplikovaných úkolů může být převedena na ekvivalentní proces s duplikovanými úkoly ale bez skrytých úkolů.

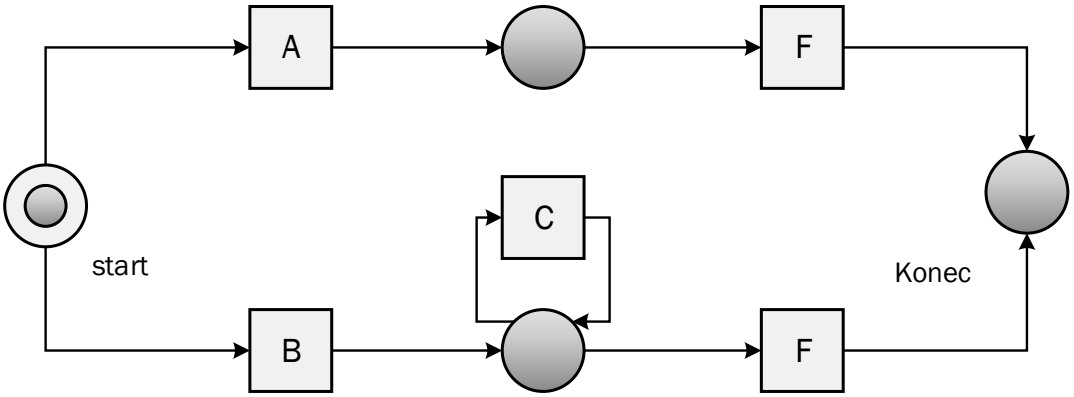


Obrázek 3.7: Model procesu s duplikovanými úkoly.

Dolování smyček

V procesu je možné, že se opakuje vykonávání stejného úkolu několikrát. Pokud toto nastane, potom se typicky jedná o smyčku v daném modelu. Obrázek 3.8 ukazuje příklad

modelu se smyčkou. Po vykonání úkolu B se může úkol C vykonat několikrát, jsou tedy možné následující sekvence BD, BCD, BCCD, BCCCD, atd. Smyčky, jako je ta na Obrázku 3.8, je lehké odhalit. Avšak smyčky se mohou v procesu vyskytovat mezi libovolnými úkoly v procesu. Pro více složité procesy není dolování smyček triviální problém, jelikož v záznamu může jeden úkol být přítomen několikrát pro danou instanci.



Obrázek 3.8: Model procesu se smyčkou.

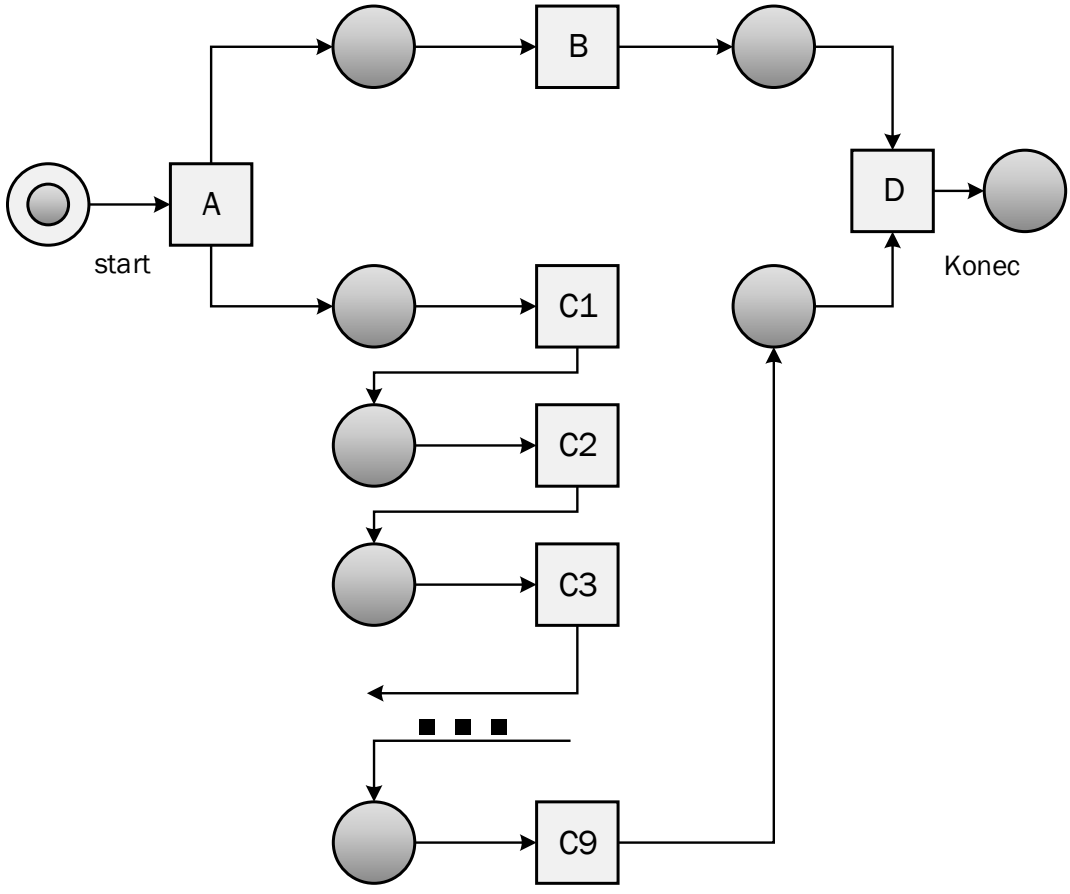
Práce s časem

Tabulka 3.4 ukazuje minimum informací potřebnou na vytvoření určité formy dolování procesu. Každý řádek popisuje událost (začátek úkolu pro specifický případ). V mnoha případech záznam obsahuje informaci o čase, takže každá událost má časové razítko. Při modelování trvání vykonání úkolu můžeme zaznamenávat začátek a konec událostí. Porovnáním odlišností mezi časovým razítkem začátku události a časovým razítkem konce korespondující události je možné určit čas zpracování dat. Informace o čase může být použita za dvěma účely: přidávání informaci o čase pro zpracování dat daného modelu a zlepšení kvality odhaleného zpracováni dat u daného modelu. Je relativně lehké rozšířit model zpracování dat o časovou informaci. Nejprve je potřeba vydolovat model zpracování dat přičemž se nebere ohled na časové známky a poté přehrát záznam v modelu zpracování dat. Přehráním záznamu je velmi jednoduché spočítat tok času, čekací dobu a čas procesů. Může ovšem nastat komplikace, že v určitých případech nalezený model procesu nemusí přesně vyhovovat dané události. Tato informace by však mohla být použita pro modifikaci modelu daného procesu (tzn. přímá modifikace výsledků z modelu, vyčištění záznamů nebo přidání nové znalosti a opakování dolovacího algoritmu). Pro zlepšení kvality záznamu se však víc využívá informace o čase. Například pokud dvě události probíhají v krátkém časovém úseku, je velice pravděpodobné, že mezi nimi bude kauzální vztah. Představa časové vzdálenosti by mohla být využita v dolovacích algoritmech. Avšak hodnota přidané veličiny dosud není známá. Fakt je, že žádná práce se tímto problémem ještě nezabývala.

Neúplnost v datech

Tento pojem souvisí s problémem *šumu*. Záznam je neúplný pokud neobsahuje dostatečné informace pro odvození procesu (Tabulka 3.4 a odvozený model procesu na Obrázku 3.5).

Předpokládejme, že na Obrázku 3.5 je správná reprezentace aktuálního procesu, avšak cesta reprezentována případem 5 je velmi neobvyklá. Pokud dolujeme jen několik instancí procesu, může se stát, že jen několik z nich, které jsou podobné případům 1,2,3,4, budou zapsány. Ve výsledku je objevený proces nesprávný, protože úkoly E a F chybí. Tento příklad může vypadat triviálně, avšak pro reálné procesy existuje až milion cest, kdy je možné povolit paralelní, podmíněné a iterativní směrování. Jako příklad uvažujme Obrázek 3.9. V tomto procesu nejsou žádná opakování, tzn. všechny příkazy jsou započaty jen jednou. Avšak, příkaz B a sekvence devíti úkolů C1, C2,. . . , C9 jsou započaty paralelně. Ve výsledku tedy existuje deset možných cest, tj. i když neexistují žádné možnosti, alespoň 10 případů musí existovat pro odvození modelu procesu, viz. Obrázek 3.9. Pozorování, kde B je započato po sekvenci 9 příkazů C1, C2,. . . , C9, je velice nepravděpodobné a pravděpodobně tisíce zaznamenaných případů jsou potřeba pro nalezení správného modelu. Pokud změníme proces na Obrázku 3.9 stejně jako příkazy C1, C2,. . . , C9, které jsou započaty paralelně, potom existuje 10!=3628800 možných cest. V tomto případě je záznam velice pravděpodobně nekompletní a je potřeba využít heuristické přístupy na vypořádání se s tímto problémem. Tyto heuristické přístupy jsou běžně založeny na principu Occamovy břitvy, tzn. princip, který popisuje: *Když existují dvě konkurenční teorie, které popisují stejné predikce, ta která je jednodušší, je lepší.*



Obrázek 3.9: Model procesu, kde je obtížné určit synchronizaci

Šum v datech

Většina algoritmů pro dolování dat předpokládá (vyžaduje), aby informace byla správná. Toto je správný předpoklad ve většině situací. Záznam může obsahovat šum tj. nesprávně zaznamenané informace. Například se může stát, že určitá situace není zaznamenána, anebo je zaznamenána až poté, co se odehrála. Algoritmus pro dolování dat by proto měl být robustní a brát ohled i na šum. Kauzální vztahy by neměli být určeny z jednoho pozorování. Algoritmus pro dolování dat by tedy měl rozlišit výjimky z normálního toku. Když je uvažován šum, je potřeba určit mezní hodnotu pro odstranění výjimek nebo nesprávných záznamů.

Shromažďování dat z heterogenních zdrojů

Dnešní podnikové informační systémy jsou komplexní a typicky složeny z obrovského počtu aplikací / komponent. Aplikace podporují fragmenty procesu a výsledkem je, že informace potřebné pro dolování procesu jsou rozptýleny přes celý podnikový informační systém. Právě proto je shromažďování zaznamenaných událostí sloužících jako vstup do dolování procesu důležité. Dokonce i v rámci jediného produktu, události mohou být zaznamenány na několika stupních v různých částech systému. Jako příklad uvažujme ERP[2] systém jako SAP[3], v němž existují desítky relevantních záznamů vhodných pro dolování procesu. Jeden přístup je použití uložených dat, které extrahují informace z těchto záznamů. V zařízeních nezávislých na XML[4] formátu se doporučuje použít jako vstup právě tento formát pro proces dolování dat.

3.4 Rozdíly u dolovacích algoritmů

V této sekci se zaměříme na rozdíly mezi dolovacími algoritmy, které jsou více popsány v článku [4]. Samozřejmě je zde silná vazba mezi dolovacím algoritmem a typem problému, na kterém lze tento algoritmus úspěšně použít. Pokud zkusíme charakterizovat dolovací algoritmy, můžeme začít výčtem typů problémů, s kterými se algoritmus dokáže bez větších obtíží vypořádat (například zvládání šumu, nekompletní záznamy, použití času, duplikované události, apod.).

Techniky pro dolování z dat se podle dosavadních zkušeností nedají použít bez modifikací na dolování procesů. To znamená, že většina technik pro dolování procesů má velice specifické vlastnosti. Navzdory tomu všemu se můžeme na dolování procesů dívat jako na podmnožinu obecného dolování dat. Mnoho charakteristik vztahujících se k obecným dolovacím algoritmům se zdají být relevantní i u dolování procesů (počáteční znalosti, lokální-globální dimenze, výpočetní složitost, nároky na paměť).

Záznamy procesů mohou obsahovat informace o atributech instancí procesů a také skutečnou cestu, kterou daná instance prošla. Pro daný model procesu se může použít tradičních dolovacích technik pro dolování rozhodovacích pravidel, které budou predikovat cestu specifické instance. Avšak při použití technik pro dolování procesů je hlavní zaměření na vytvoření modelu, ne na indukci pravidel pro predikci cesty grafem pro jednu instanci.

[2] Enterprise Resource Planning, podnikový informační systém, je označení systému, jímž podnik (nebo jiná organizace) za pomoci počítače řídí a integruje všechny nebo většinu oblastí své činnosti, jako jsou plánování, zásoby, nákup, prodej, marketing, finance, personalistika, atd.

[3] Systems - Applications - Products in data processing

[4] Extensible Markup Language

Většina záznamů událostí obsahuje pouze pozitivní příklady. Pokud obsahuje i šum, je možné, že obsahuje i negativní příklady. U těchto příkladů však chybí nějaký atribut, který by řekl, že jsou negativní. Pouze pokud by byl přítomný specialista na daný proces, je možné, že by dokázal určit negativní příklady (tato instance procesu nemůže nikdy nastat).

Počáteční znalosti dolování procesů

Dolování dat i dolování procesů je vlastně hledání ve velkém prostoru možných modelů, které jsou implicitně definované jazykem, který používáme pro reprezentaci objevených modelů. Cílem takového hledání je nalezení modelu, který se nejvíce blíží datům ze záznamu událostí. Výběr jazyka pro realizaci procesu silně ovlivňuje dolovací proces. Příkladem takového jazyka může být *Petriho síť*, *blokově orientované procesní modely* (block-oriented process models) nebo *diagram procesu řízeného událostmi* (event dependency models). Některé modelovací jazyky jsou robustnější než jiné. Například blokově orientované modely je možné lehce převést na Petriho sítě, ale opačný převod není vždy možný. Z tohoto důvodu je Petriho síť robustnější reprezentativní jazyk než blokově orientované modely.

Pokud nevíme nic o procesu, který chceme dolovat, je nejvhodnější použít nejrobustnější reprezentativní jazyk modelu procesů, který je k dispozici. Neboť výběr méně robustního jazyka by nám mohl znemožnit najít odpovídající model. Avšak výběr nejrobustnějšího jazyka má také svůj negativní efekt, jelikož nám poroste velikost prostoru, ve kterém budeme hledat. Velký vyhledávací prostor zvýší šum u dolovací techniky, potřebuje více dat pro úspěšné dolování a má negativní efekt na výpočetní složitost a nároky na paměť. Tato situace se velice podobá situaci, kdy hledáme správný regresivní model. Pokud víme, že hledáme lineární model, tak použijeme lineární regresi jako modelovací techniku. Poté nám stačí menší množství dat, zvolený přístup je méně náchylný na šum a výpočetní čas je kratší, než kdybychom použili nelineární model. Pokud tedy dopředu víme, jaký typ procesu budeme modelovat a použijeme tuto informaci k vhodnějšímu zvolení reprezentativního jazyka, poté můžeme říct, že máme velké počáteční znalosti. V praxi se nejvíce používají dolovací algoritmy, u kterých máme velké počáteční znalosti.

Lokálně – globální dimenze

Známe-li jazyk pro reprezentaci modelu procesu, můžeme se na dolování dívat jako na hledání jednoho možného řešení z velkého prostoru kandidátů. Dolovací algoritmy mohou využívat různé strategie, aby našly nejvhodnější model. Rozlišujeme dvě základní strategie, a sice lokální strategie založené na postupném sestavování optimálního modelu daného procesu z úzce lokálních informací a globální strategie založené na přímém hledání optimálního modelu.

Příkladem úzce lokálních strategií jsou α – algoritmy (tento algoritmus je více popsán v knize [1]) a dolovací techniky založené na Markovovském přístupu. V těchto postupech se využívají pouze úzce lokální informace o binárních vztazích mezi událostmi. Příkladem pro globální strategie může být například hledání optimálního modelu v genetice. Genetické hledání začíná u populace, která obsahuje kompletní modely procesu. Proces je globální, protože kvalita vhodnosti vybraného procesu se počítá mezi modelem procesu a všemi možnými cestami v toku procesu zaznamenaných událostí.

Oba dva přístupy mají své výhody a nevýhody. Všeobecně, lokální strategie jsou méně komplexní z výpočetního hlediska a požadavky na paměť jsou nižší než u globálních strategií. Avšak neexistuje žádná garance, že výstupy z lokální strategie (na stupni binárních

vztahů mezi událostmi) budou vyplývat z globálního optimálního modelu procesu. Takže interpretace lokálních dolovacích technik může být chybná, pokud potřebné lokální informace nejsou dostupné.

Například α – algoritmus zmíněný výše nedokáže pracovat s konstrukcemi, které nejsou libovolně volitelné, protože výběr mezi úkoly není definovaný uvnitř nějakého uzlu, ale může záviset na výběrech z jiných části modelu procesu. U globálních technik je větší šance na nalezení konstrukce ne libovolně volitelné. V praxi je velice neobvyklé, aby záznamy byly kompletní nebo bez šumu. Proto je pak důležité, jak citlivý je algoritmus na šum. Základním problémem je, jestli jediný chybný příklad může obrátit vzhůru nohama odvození správného modelu. Globální strategie jsou většinou méně náchylné na šum. V některých případech je možno kombinovat lokální strategie s globálními. Nejprve se použije lokální přistup, a poté je pomocí globálního přístupu provedena kontrola na celý model. V případě nedostatků je model automaticky aktualizován nebo jsou podány návrhy, jak model opravit.

Kapitola 4

Popis použitých dolovacích metod

V této práci jsou k dispozici data od výrobní továrny, jejíž výrobní proces je znám. Proto jsou metody použité v návrhové a implementační části zaměřeny na *vylepšení procesu* (process enhancement) a na *perspektivu instancí procesu* a *časovou perspektivu*. Při technice *vylepšení procesu* se mohou využívat metody používané v klasickém získávání znalostí z databází. Proto bude v této kapitole vysvětlen pojem frekventovaná množina a algoritmus pro získavaní frekventovaných množin - *Frequent Pattern Growth*, neboť na jejich základě se provádí dvě ze tří zvolených dolovacích technik. V závěru kapitoly bude nastíněn princip klasifikace na základě asociačních pravidel, jelikož je tato metoda použita u třetí dolovací úlohy.

4.1 Frekventovaná množina

Získávání frekventovaných množin (nebo také frekventovaný vzor) je jednou z nejvýznamějších technich pro charakterizaci dat. Tento problém se někdy zaměňuje za získávání asociačních pravidel, avšak asociační pravidla jsou více komplexní charakteristikou dat a jejich získání je těsně spjato se získáním frekventovaných množin. Při psání této části jsem vycházel z [6].

Intuitivně, množině položek, které se často vyskytují spolu, můžeme říkat frekvetovaná množina. Nejčastěji se tyto množiny dolují z transakčních databází, tj. databáze kde jeden řádek tabulky představuje jednu transakci. Avšak v našem případě jsou data uloženy v klasické relační tabulce. Proto bude potřeba menší modifikace algoritmu.

Abychom mohli definovat frekventovanou množinu, potřebujeme nějaké číslo p, kterému se říka *minimální podpora*. Pokud M je množina položek, poté *podpora* pro M je počet transakcí, ve kterých je M podmnožinou. Množinu M nazveme frekventovanou, pokud je její podpora rovna nebo je větší než zadaná minimální podpora p. Jednu takovou transakční databazi máme v Tabulce 4.1 a na jejím příkladu si ukážeme frekventované množiny.

Jelikož je prázdná množina podmnožinou všeh množin, podpora pro $\emptyset$ je 8. Avšak prázdnou množinou se všeobecně není třeba zabývat, neboť nám nepřináší žádné znalosti.

Mezi jednoprvkovými množina je zřejmé, že $\{pes\}$ a $\{krtek\}$ jsou poměrně časté. *Pes* se vyskytuje ve všech transakcích kromě transakce číslo 5, takže jeho podpora je 7. *Krtek* se vyskytuje v šesti z osmi transakcí. Chybí v čísle 4 a 8, takže má podporu 6. *Losos* se vyskytuje také poměrně často, patří do transakcí 1, 2, 5, 7 a 8, takže jeho podpora je 5. Dvě zvířáta *orel* a *slon* jsou ve třech transakcích, a dále *kůň* a *klokan* pouze ve dvou. Žádné další zvíře se nevyskytuje více jak jednou.

Transakce	Položky v transakci
1	krtek, orel, pes, žirafa
2	gepard, tygr, lev, krtek, pes, losos, hroch, želva, ptakopysk, orel
3	krtek, pelikán, muflon, pes, lama, losos
4	jaguár, levhart, gorila, pes, albatros, činčila, slon
5	krtek, orel, buvol, býk, slon, žralok
6	pes, krtek, slon, prase, krává
7	pes, orel, krtek, losos, kapr, zebra
8	kůň, bizon, pelikán, pes, kobra, orel, pavouk

Tabulka 4.1: Tabulka transakcí

Dále budeme počítat s minimální podporu, která je nastavena na hodnotu $p = 3$. Poté máme pět jednoprvkových frekventovaných množin: $\{pes\}$, $\{krtek\}$, $\{losos\}$, $\{orel\}$ a $\{slon\}$.

Nyní se podíváme na dvouprvkové množiny. Dvouprvková množina nemůže být frekventovaná, pokud obě položky v množině nejsou frekventované samy o sobě. Z toho nám vyplývá, že je možných jen 10 dvouprvkových množin. Tabulka 4.2 ukazuje všechny možné dvouprvkové množiny.

	slon	**losos**	**orel**	**krtek**
pes	4,6	2,3,7	1,2,8	1,2,3,6,7
krtek	5,6	2,3,7	1,2,5	
orel	5	2,7		
losos	žadný			

Tabulka 4.2: Tabulka výskytů dvouprvkových množin

Například z tabulky 4.2 mějme dvojici $\{pes, slon\}$. Tato množina se vyskytuje pouze v transakcích 4 a 6, tím pádem je její podpora 2. Pokud máme minimální podporu rovnu třem, poté mezi frekventované dvouprvkové množiny patří 5 množin:

$$\{pes, losos\}\{pes, orel\}\{pes, krtek\}$$
$$\{krtek, losos\}\{krtek, orel\}$$

Každá množina se vyskytuje v transakční tabulce 4.1 třikrát, jenom množina $\{pes, krtek\}$ se vystkytuje čtyřikrát.

Za další se podíváme na tříprvkové množiny. Aby byla tříprvková množina frekventovaná, musí všechny páry položek množiny tvořit frekventované dvouprvkové množiny. Například $\{pes, losos, orel\}$ nemůže být frekventovaná, protože kdyby byla, tak musí být frekventovaná množina $\{losos, orel\}$, ale ta není. Trojice $\{pes, krtek, orel\}$ by mohla být frekventovaná, protože všechny dvouprvkové podmnožiny jsou frekventované. Bohužel, tato trojice se v transakční tabulce vyskytuje pouze dvakrát, a to na řádcích 1 a 2, tím pádem nemůže být frekventovaná. Avšak trojice $\{pes, krtek, losos\}$ je frekventovaná, neboť všechny dvojice jsou frekventované a samotná trojice se v tabulce nachází třikrát na řádcích 2, 3 a 7. Žádná další trojice není frekventovaná. Jelikož máme pouze jednu trojici frekventovanou, nemůžeme mít žádnou frekventovanou čtveřici či větší množiny.

Formální definice

Předpokládejme, že máme relační tabulku R, která je definována doménami $D_1, D_2, \ldots, D_n$. Relační tabulka je potom definována jako dvojce $R = (H, R^*)$, kde H je hlavička relační tabulky a R^* je tělo tabulky, které obsahuje samotné záznamy. Hlavička tabulky je definována jako množina $H = \{(A_1 : D_1), (A_2 : D_2, \ldots, (A_n : D_n)\}$, kde $A_i \neq A_j$ pro každé $i \neq j$. A_i, pro každé $i = 1, 2, \ldots, n$, jsou atributy tabulky a D_i jsou korespondující domény (množiny možných skalárních hodnot atributů. které musí být tejného typu). Tělo R^* relační tabulky je definovano jako relace $R \subseteq D_1 \times D_2 \times \cdots \times D_n$

Pokud je doména D_i pro atribut A_i konečná, potom je atribut kategorický. Naopak, pokud je doména nekonečná a mezi atributy je definováno uspořádání, poté je atribut nazýván kvantitativním nebo numerickým.

Cílem je najít množinu frekventovaných množin v relační tabulce. Frekventovaná množina FM v relační tabulce je definována jako množina predikátů ve tvaru $\{a_1, a_2, \ldots, a_n\}$. Tato množina predikátů a jejich hodnot musí odpovídat počtu řádků z těla relační tabulky R^*. Množina FM všech frekventovaných množin se poté může zapsat jako:

$$FM = \{fm \mid podpora(fm) \geq minPodpora\}$$

Výsledná hodnota podpory p pro množinu predikátů P je definována jako poměr řádků tabulky, ve kterých jsou hodnoty z řádku stejné jako hodnoty z množiny predikátů P, s celkovým počtem řádků v relační tabulce. Tento poměr může být vyjádřen pomocí rovnice jako:

$$P(fm) = \frac{n}{N},$$

kde N je celkový počet řádků v relační tabulce R a n je počet řádků, které odpovídají množině hodnot P.

Pokud má množina hodnot P podporu stejnou nebo větší nez je zadaná minimální podpora, potom:

$$s = \{a_1, a_2, \ldots, a_n\} \mid \exists\, Radky = \{r_i \mid r_i \in R \wedge \forall a_k \in s : a_k \approx r_i, k = 1 \ldots n\} : \frac{|Radky|}{N} \geq minPodpora$$

Výraz $a_k \approx r_i$ značí to, že hodnoty obsažené v predikátu a_k jsou také obsaženy v řádku r_i z relační tabulky R.

Pokud je atribut A kategorický, tedy *booleovského typu* nebo nabývá numerikých hodnot s malým oborem hodnot, je definován jako:

$$(A = v) \approx r_i \Leftrightarrow \exists\, j : A_j = A \wedge v_{ij} = v$$

Dále může být atribut A kvantitativní s velkým oborem hodnot, poté je definován jako:

$$(A = |l, h|) \approx r_i \Leftrightarrow \exists\, j : A_j = A \wedge l \leq v_{ij} \leq h \tag{4.1}$$

Rovnice 4.1 vede k problému diskretizace kvantitativních atributů, neboť je nutné znát hodnoty l a h.

4.2 Algoritmus Frequent Pattern Growth

Algoritmus Frequent Pattern Growth (dále jen *FP Growth*) je jedním ze dvou algoritmů pro získávání frekventovaných množin. Poprvé byl tento algoritmus popsán v článku [11]. Dalším je *apriori* algoritmus, který je popsán v článku [5]. V této práci byl vybrán algoritmus *FP Growth* kvůli své rychlosti a dalším vlastnostem, které budou zmíněny dále. V krátkém shrnutí si ukážeme, v čem tkví nevýhoda použití *apriori* algoritmu:

- Apriori algoritmus je velmi nákladný na pamět a procesorový čas, jelikož při dolování generuje velké množství kandidátů na frekventované množiny,. Například, pokud máme 10^4 jednoprvkových frekventovaných množin, tento algoritmus musí generovat více jak 10^7 dvouprvkových kandidátů. A dále u každého kandidáta musí zjistit jejich frekvence výskytu v databázi. Za další, pro nalezení frekventovaného vzoru velikoti 100, tedy $\{a_1, a_2, \ldots, a_{100}\}$, musí algoritmus vygenerovat přibližně $2^{100} \approx 10^{30}$ kandidátů. Toto je vrozená cena generování kandidátů, nehledě na použité heuristiky a vylepšení algoritmu.

- Je velmi zdlouhávé opakovaně skenovat databázi a porovnávat množinu kandidátů se vzory. Toto je zvláště pravdivé pro dlouhé vzory.

V následujících podkapitolách budou blíže rozebrány tři základní prvky algoritmu *FP Growth*. Nejprve se popíše *FP strom*, dále je popsána tvorba tohoto stromu a nakonec technika pro dolování frekventovaných vzorů.

FP strom (Frequent Pattern Tree)

FP strom je kompaktní datová struktura, která je rozšířením prefixového stromu. Ukládá v sobě důležité kvantitativní informace o frekventovaných vzorech. Pouze frekventované vzory délky jedna jsou obsaženy v tomto stromě. Stromové uzly jsou uspořádány tak, že více frekventované uzly mají větší šanci na sdílení uzlů než méně frekventované uzly.

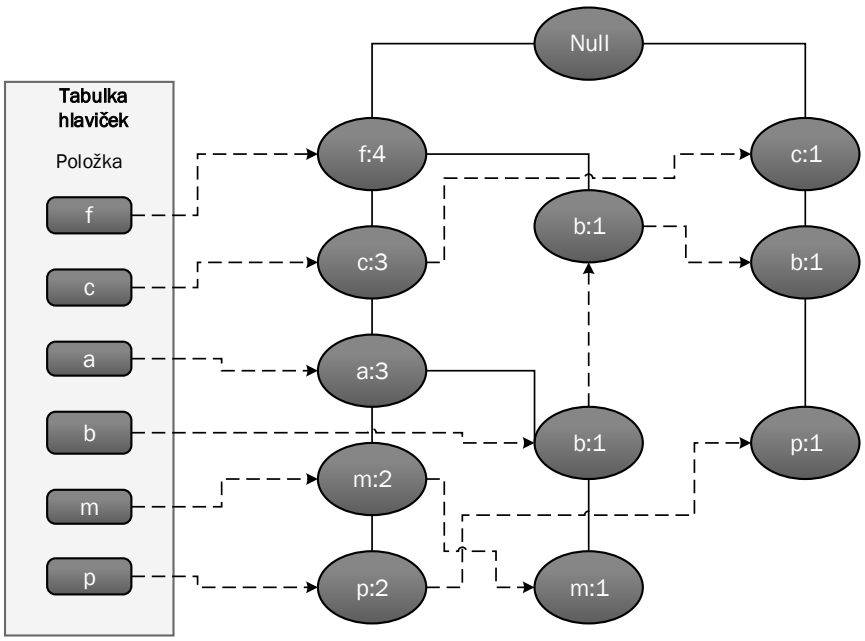
Nejprve si ukážeme FP strom na příkladu a poté si ho formálně definujeme. Mějme transakční tabulku 4.3 a minimální podporu zvolíme $p = 3$. V prvním kroku se naskenuje databáze, tedy naše transakční tabulka, a zjistí se seznam frekventovaných položek. Výsledkem je: $\langle (f : 4), (c : 4), (a : 3), (b, 3), (m : 3), (p : 3) \rangle$. Tento seznam je seřazen podle hodnoty podpory, která je u každé položky vyjádřena za znakem „:“ . Seřazení je velmi důležité, neboť každá cesta stromem bude dbát na toto seřazení. Frekventované množiny v každé transakci v Tabulce 4.3 již jsou sežazeny podle hodnoty podpory v nejpravějším sloupci.

ID transakce	Položky v transakci	Seřazený seznam položek
100	f ; a; c; d; g; i; m; p	f ; c; a; m; p
200	a; b; c; f ; l; m; o	f ; c; a; b; m
300	b; f ; h; j; o	f; b
400	b; c; k; s; p	c; b; p
500	a; f ; c; e; l; p; m; n	f ; c; a; m; p

Tabulka 4.3: Transakční databáze pro ukázkový příklad

V druhém kroku se vytvoří kořen stromu, který se bude značit *null*. Znovu se prochází databáze. Z první transakce je vytořena první větev stromu:
$\langle (f:1), (c:1), (a:1), (m:1), (p:1) \rangle$. Všimněme si, že tyto frekventované položky jsou seřazeny stejně jako položky v transakci. Frekventované položky druhé transakce $\langle f, c, a, m, b \rangle$ mají stejný prefix $\langle f, c, a \rangle$ jako již existující cesta $\langle (f, c, a, m, p \rangle$ ve stromu . Hodnota u každého uzlu ve stromě, které jsou v prefixu transakce, jsou inkrementovány o jedničku. Dále je vytvořen nový uzel $(b:1)$ a je spojen jako potomek s uzlem $(a:2)$. Podobně je vytořen uzel $(m:1)$ a je spojen jako potomek s uzlem $(b:1)$. Frekventované položky třetí transakce $\langle f, b \rangle$ sdíli pouze uzel $\langle f \rangle$ s *f-prefixovým podstromem*. Tím pádem je inkrementována hodnota u uzlu $\langle f \rangle$ a je vytvořen nový uzel $(b:1)$, který je spojen jako potomek s uzlem $(f:3)$. Zpracováním čtvrté transakce dojde k vytvoření druhé větve ve stromě: $\langle (c:1), (b:1), (p:1) \rangle$. Poslední transakce obsahuje seznam frekventovaných položek $\langle f, c, a, m, p \rangle$, který je stejný jako u první transakce. Tím pádem jsou o jedničku inkrementovány hodnoty všech uzlů podél již existující cesty $\langle f, c, a, m, p \rangle$.

K usnadnění průchodu stromem je vytvořena tabulka hlaviček, ve které každá položka ukazuje na svůj výskyt ve stromě. Uzly se stejným jménem jsou spolu také propojeny a tvoří sekvenci. Ukázka výsledného stromu je vidět na Obrázku 4.1



Obrázek 4.1: FP strom pro představený příklad

FP strom se dá definovat jako trojice pravidel:

1. Skládá se z kořene, který se značí *null*, dále z množiny prefixových podstromů tvořící potomky kořene a z *tabulky hlaviček frekventovaných položek*.

2. Každý uzel v prefixovém podstromě je tvořen třemi prvky: *jménem položky, hodnotou a spojením*. *Jméno položky* značí, ke které položce uzel patří. *Hodnota* reprezentuje počet transakcí, ve kterých se vyskytuje část cesty až po konkrétní uzel. *Spojení* tvoří

ukazatel na další uzel ve stromě se stejným jménem položky. *Spojení* neexistuje, pokud ve stromě není další položka stejného jména.

3. Každý záznam v *tabulce hlaviček frekventovaných položek* se skládá ze dvou prvků: (1) *jméno položky* a (2) *hlavičky spojení*, která ukazuje na první uzel v FP stromě se stejným *jménem položky*.

Na základě této definice máme následujcií algoritmus pro konstrukci FP stromu:

Algoritmus pro konstrukci FP stromu

Vstup: Transakční databáze DB a minimální podpora p

Výstup: FP strom (frequent Pattern Tree)

Metoda: FP strom je zkonstruován v následujících krocích:

1. Jednou projdi transakční databázi DB a vytvoř množinu frekventovaných položek F spolu s jejich *podporou*. Seřaď F sestupně podle podpory jako S – seznam frekventovaných položek.

2. Vytvoř kořen FP stromu K a nazvi ho *null*. Pro každou transakci v databázi vykonej následující:

   Vyber a seřaď frekventované položky v transakci podle pořadí v seznamu S. Nechť je seřazený seznam frekventovaných položek v transakci značen $[p|P]$, kde p je první element a P je zbývající seznam. Zavolej metodu $vlozDoStromu([p|P], K)$.

   Metoda $vlozDoStromu([p|P], T)$ je prováděna následovně:
   Pokud K má potomka U takového, že $U.jménoPoložky = p.jménoPoložky$, poté inkrementuj hodnotu potomka U o 1;
   Jinak vytvoř nový uzel U. Z uzlu U vytvoř potomka uzlu T a dále spoj uzel U s dalšími uzly se stejným jménem. Pokud není P prázdné, volej metodu $vlozDoStromu(P, U)$ rekurzivně

Z konstrukce *FP stromu* můžeme vidět, že je potřeba přesně dvě čtení z databáze. Poprvé se vytvoří množina frekventovaných položek, a při druhém průchodu se vytvoří celý *FP-strom*. Cena vkládání jedné transakce do *FP stromu* je $O(|Transakce|)$, kde $|Transakce|$ je počet obsažených frekventovaných položek. Výsledný FP strom je kompletní a kompaktní.

Algoritmus pro dolování frekventovaných vzorů z FP stromu

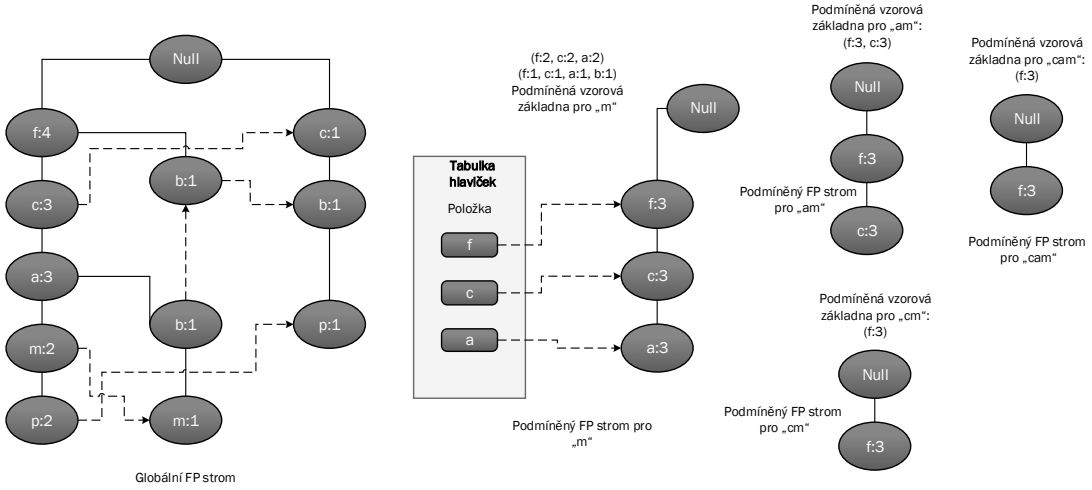
Konstrukce kompaktního *FP stromu* zajišťuje, že dolování může být provedeno s kompaktní datovou strukturou. Avšak to automaticky negarantuje vysokou efektivnost, protože se musí počítat s kombinatorickým problémem generování kandidátů, pokud použijeme *FP strom* k dolování a zkontrolování všech možných frekventovaných vzorů.

Dále bude ukázán příklad na dolování frekventovaných množin z jedné větvě stromu a poté pseudokód algoritmu pro získání kompletní množiny frekventovaných vzorů. Příklad je převzat z článku [11], kde jsou i více rozebrány vlastnosti tohoto algoritmu. Příklad dolování je ukázán na *FP stromě*, který je na zobrazen na Obrázku 4.1. Dolování začína od posledního prvku v hlavičce uzlů.

Pro uzel p máme frekventovanou množinu *(p:3)* a dvě cesty v *FP stromě*: $\langle f:4, c:3, a:3, m:2, p:2\rangle$ a $\langle c:1, b:1, p:1\rangle$. První cesta značí, že se transakce *(f, c, a, m, p)* vyskytuje dvakrát v databázi. Transakce *(f, c, a)* se vyskytuje třikrát a samotné *(f)* čtyřikrát. Avšak dohromady s p se vyskytují pouze dvakrát. Tím pádem pro zjištění, které prvky se vyskytují spolu s p, nám stačí pouze prefixová cesta k p $\langle f:4, c:3, a:3, m:2\rangle$. Podobně i druhá cesta *(c, b, p)* se v databázi vyskytuje jednou a prefixová cesta pro p je $\langle c:1, b:1\rangle$. Dvě zmíněné prefixové cesty tvoří *podmíněnou základu pro p*. Konstrukce *FP stromu* na této podmíněné základně vede pouze k jedné větvi $(c:3)$. Tím pádem je získán pouze jeden frekventovaný vzor $(cp:3)$. Hledaní frekventovaných vzorů spojených s p je ukončeno.

Pro uzel m je odvozen frekventovaný vzor $(m:3)$ a dvě cesty $\langle f:3, c:3, a:3, m:2\rangle$ a $\langle f:3, c:3, a:3, b:1, m:1\rangle$. P se vyskytuje společně s m také, avšak již není potřeba ho zde uvádět, neboť všechny frekventované množiny s p již byly nalezeny. Obdobně jako předchozí analýza i zde jsou vytvořeny dvě podmíněné základny $(f:2, c:2, a:2)$ a $(f:1, c:1a:1, b:1)$. Konstrukcí *FP stromu* získáme podmíněný *FP strom* $\langle f:3, c:3, a:3\rangle$, který obsahuje pouze jednu cestu frekventovaných položek. Poté je možné zavolat dolování *FP podmíněného stromu* rekurzivně.

Obrázek 4.2 ukazuje výše zmíněné rekurzivní dolování, které zahrnuje získání tří položek *(a)*, *(c)* a *(f)* v sekvenci. Po ukončení dolování všech podstromů jsou výsledné množiny následující: $(m:3), (am:3), (cm:3), (fm:3), (cam:3), (fam:3), (fcam:3), (fcm:3)$. Dolování obdobně pokračuje pro všechny položky v hlavičce.



Obrázek 4.2: Konstrukce podmíněných FP stromů, značí se *FP strom*$|m$

Pseudokód algoritmu Frequent Pattern Growth

Vstup: *FP strom* sestavený podle výše uvedeného algoritmu

Výstup: Kompletní množina frekventovaných vzorů

Metoda: Volej funkci *FPGrowth(FP strom, null)*.

Metoda FProwth(Strom, α)
{
(1) **if** Strom obsahuje jednu cestu C
(2) **then for each** kombinaci uzlů v cestě P (označenou jako β)
(3) generuj vzor $\beta \cup \alpha$ s *podporou* = *minimální podpora uzlů v* β;
(4) **else for each** a_i v hlavičce *Stromu* do {
(5) generuj vzor $\beta = a_i \cup \alpha$ s *podporou* = a_i.*podpora*;
(6) zkonstruuj β *podmíněnou vzorovou základnu*
a potom β *podmíněný FP strom* $Strom_\beta$;
(7) **if** $Strom_\beta \neq \emptyset$
(8) **then** zavolej metodu $FPGrowth(Strom_\beta, \beta)$ }
}

4.3 Klasifikace na základě asociačních pravidel

Asociační pravidla byla blíže popsána v Kapitole 2.3, proto si v této sekci ukážeme pouze princip klasifikace, která tato asociační pravidla využívá. Metoda, která zde bude popsána vychází z [7], kde byla klasifikace použita na textové dokumenty.

Textové dokumenty jsou reprezentovány jako transakce, které obsahují množiny slov vyskytujících se v dokumentu. V trénovací fázi jsou nalezeny asociační pravidla pro každou třídu zvlášť pomocí algoritmu *apriori*. Pro klasifikaci v této práci jsou potřeba pouze asociační pravidla ve specifické formě. Požadovaný tvar pravidla, které chceme získat během trénovací fáze má tvar:

$$slovo_1 \wedge slovo_2 \wedge \ldots \wedge \Rightarrow Category,$$

kde část před implikací obsahuje množinu slov, které se často vyskytují spolu v dokumentu (frekventovaná množina) a náleží do kategorie, která je vyjádřena v části za implikací. Můžeme vidět, že úlohou trénovací fáze je získat množinu asociačních pravidel pro každou kategorii. Tyto množiny poté tvoří klasifikátor.

Množina kategorií je pevně daná spolu s dokumenty, které do každé kategorie náleží. Díky tomu můžeme množinu asociačních pravidel získat pro každou třídu odděleně. Nejprve se tedy naleznou frekvetované množiny pro každou kategorii a poté se tyto množiny spojí s odpovídající kategorií do asociačního pravidla. Díky tomu se tato metoda označuje *ARC-BC* (Association Rule Classifiation - By Category).

Jednou z vlastností je i možnost klasifikovat do více tříd, neboť může existovat asociační pravidlo, které má stejnou část před implikací, ale různou kategorii za implikací. Pokud je nezbytně nutné mít pro jedno asociační pravidlo pouze jednu kategorii, poté se musí rozhodnout na základě podpory nebo spolehlivosti, které pravidlo je ponecháno. Nejčastěji se odstraní asociační pravidlo s menší podporou nebo spolehlivostí.

Získaná množina pravidel často obsahuje velký počet stejných asociačních pravidel. K snížení tohoto počtu se mohou použít *ořezávací techniky* (pruning techniques). Úkolem je nalézt taková asociační pravidla, která jsou více obecná a mají větší hodnotu spolehlivosti.

Jakmile se získá vhodný počet pravidel pro každou kategorii, může se přistoupit k přiřazení třídy objektu (dokumentu). Nejprve se získá množina slov reprezentující dokument a poté se tato množina porovná s asociačními pravidly. Pokud existuje jedno pravidlo, objektu se přiřadí odpovídající kategorie.

Obvykle se nalezne více jak jedno asociační pravidlo (mezi kategoriemi) pro klasifikaci dokumentu. Je nezbytné definovat *dominantní faktor*, který je vypočítán jako suma spolehlivosti jednotlivých pravidel v kategorii. Díky tomu je možné získat nejvíce domintantí kategorii, nebo k dominantních kategorií.

Kapitola 5

Analýza procesu a návrh aplikace pro dolování

V této kapitole se analyzuje výrobní proces a jsou představeny možné kroky ve fazi předzpracování dat. V následujících podkapitolách jsou popsány principy zvolených metod dolování procesů. První zvolená metoda je samotné *objevení procesu* (process discovery). Další metoda se snaží zjistit frekventované množiny při rychlém nárustu front před jednotlivými stroji v procesu. Poslední metoda je klasifikace a predikce produktu na základě frekventovaných množin před vstupem do jednoho úkolu. V závěru kapitoly jsou navrženy specifikace požadavků a je ukázán grafický návrh výsledné aplikace. Při analýze a návrhu dolovacích metod jsem se inspiroval v [8], [14] a [10].

5.1 Analýza výrobního procesu

Data k této práci byla anonymizována, tudíž se dále budeme bavit o společnosti, která vyrábí *předmět*. Každý předmět je charakterizován několika atributy (28 různých atributů) a na základě těchto atributů jsou vykonávány různé úlohy. Každý předmět má dále svoje ID, tudíž může být brán jako jedna instance procesu při dolování. Předmět je zpracováván v různých strojích (události). Některé stroje pracují paralelně a některé stroje jsou spojeny s více úkoly. Stroje jsou brány jako *zdroje*, které mohou být vypnuty nebo mohou pracovat. Lidé pracují na stojích nebo na jiných manuálních pracovištích. Pravděpodobnost cesty je přesná na 100 %, jelikož *předmět* s konkrétními atributy musí být zpracován konkrétním strojem s přesně daným nastavením.

Výrobní zdroje pro předmět jsou celkem předvídatelné, jelikož jsou plánovány několik dní dopředu. Jediným neznámým parametrem jsou časy vykonání každého úkolu, který je závislý na atributech *předmětu*. Každá instance procesu představuje zpracování jednoho *předmětu*, tím pádem atributy instance jsou atributy *předmětu*. Tyto atributy jsou známy na začátku výrobního procesu, takže není nutné dělat zmeny v dolování v průběhu zpracování. Časy vykonání jsou závislé nejen na parametrech *předmětu*, ale i na lidech obsluhující daný stroj a na frontě před strojem.

Data pro získávání znalostí jsou rozděleňa do dvou souborů. První soubor obsahuje záznam událostí, který byl získán v časovém horizontu 8 měsíců. Každý řádek v záznamu událostí představuje jeden úkol *předmětu* na jednom stroji. Druhý soubor obsahuje atributy jednotlivých předmětů a každý *předmět* je identifikován pomocí ID.

5.2 Předzpracování dat

Každé dolování dat potřebuje čištění dat a předzpracování. Tato fáze je nejvíce závislá na typu dat.

Analýza kvality dat

Prvním krokem při dolování dat je získat informaci o spolehlivosti dat. Musíme analyzovat záznamy se stejnými atributy. Poté musíme spočítat odchylku těchto záznamů. Například pokud máme 20 záznamů s atributy A, B, C, spočítáme jejich odchylku času provedení. Pokud výsledná odchylka je příliš vysoká, jsou data k dolování nepoužitelná. Znamená to, že se na atributy nemůžeme spolehnout, nebo některé atributy chybí, anebo při měření nastala chyba (poslední problém bývá nejčastější).

Nízké a vysoké hodnoty

V datech se občas vyskytují extrémní hodnoty. Velice nízké hodnoty jsou pravděpodobně chyby na předmětu, které byly objeveny, a následně byl předmět poslán na opravu. Vysoké hodnoty jsou poruchy nebo špatně změřené hodnoty. Musíme si dávat pozor, neboť přestávka může ovlivnit čas zpracování. Předpokládejme, že startovní čas byl změřen a uložen korektně, poté nastala přestávka a poté se proces výroby znovu rozeběhl. Celkový čas úkolu je nepoužitelný. Pokud bychom takové chyby a poruchy chtěli analyzovat, potřebovali bychom o nich informaci v datech nebo nějaký rozpis přestávek nebo poruch.

Nekompletní měření

Někdy se stane, že je k dispozici pouze informace o začátku nebo konci úkolu. Je možné tento chybějící čas odvodit. Odvození vychází z toho, že pokud produkt A vstoupí do úkolu a je zaznamenán startovní čas a produkt B vstoupí do stejného úkolu a je u něj také zaznamenán startovní čas. Konečný čas produktu A v daném úkolu spočítáme jednoduše rozdílem $start(B) - start(A)$. Tento přístup je jednoduchý, ale nemusí vždy fungovat. Pokud počítáme čas (startovní nebo koncový), tak daný úkol musí mít vysoké využití (krátká čekací doba na další produkt). Každé čekání se započítává do výsledného času na zpracování, neboť nevíme, jestli se produkt zpracovával nebo se čekalo na další. Jiným řešením tohoto problému je vypočítat čas na zpracování z jiných produktů. Pokud máme k dispozici produkty s velmi podobnými atributy a jejich časy jsou známé a odchylka je nízká, poté dokážeme odvodit čas i u produktu, kterému jeden z časů chybí.

Změny v čace

Reálné procesy nejsou statické. Časy vykonání jednotlivých úkolů se mohou měnit. Tento problém nemusí být způsoben samotným procesem. Mohou nastat neočekáváné události, které se projeví v datech o procesu, ale které mohou být způsobeny vnějšími faktory. Například dělník obsluhující stroj si mohl vzít přestávku, na stroji doško k poruše apod. Tyto události by němely být zapsány do záznamu událostí. Pokud se však zapisují všechny události a na vnější faktory se nebere ohled, měly by se ve fázi předzpracování tyto odchylky odstranit.

5.3 Návrh dolovacích metod

Ukázka procesu pomocí grafu

Jak bylo ukázáno v teorii o dolování procesů, jedním typem dolování je objevení procesu. Proces výroby je přesně daný a tudíž by nebylo potřeba takové dolování provádět. Ale jelikož byla data anonymizována a já neměl přístup k jiným datům o výrobní továrně a jejím procesu, rozhodl jsem se tento typ dolování zahrnout i do výsledné aplikace.

Cílem je nalézt graf všech strojů a jejich propojení mezi sebou. Výsledný graf by tedy měl přesně reprezentovat výrobní linku továrny. Bohužel nemám k dispozici reálný proces výroby továrny, tudíž není možné porovnat výsledný model procesu s odpovídající realitou. Výsledný graf může být sestaven ze dvou typů uzlů. Buď jako uzly grafu mohou být jednotlivé úkoly na strojích, nebo to mohou být jednotlivé stroje. Z analýzy procesu je již známo, že jeden stroj může vykonávat více úkolů. Po prozkoumání dat bylo zjištěno, že počet všech úkolů je 35 a počet všech strojů je 20.

Ve výsledné aplikaci je zobrazen graf, kde uzly představují stroje. Menší počet strojů nám zajistí lepší přehlednost grafu. Přece jen 35 uzlů by bylo více nepřehledných, a to ještě není zmíněn počet možných hran mezi uzly, který s počtem uzlů přímo souvisí. Avšak případná modifikace algoritmu tak, aby zobrazoval uzly představující jednotlivé úkoly na strojích, by nebyla nemožná.

Analýza procesu na základě frekventovaných množin

Cílem této metody je zjistit frekventované množiny atributů produktu, které nějakým způsobem napomohly ke zpomalení výrobního procesu. Data pro analýzu jsou získána pomocí simulátoru produkční historie, který jako vstup bere záznam událostí. Simulátor prochází záznam událostí a zjišťuje délky front před jednotlivými stroji. Pokud fronta před nějakým strojem narostla příliš rychle během daného časového intervalu, pak jsou produkty a jejich atributy uloženy do datové sady a následně použity pro analýzu.

Délka fronty není nejdůležitějším kritériem pro uložení informace, protože dlouhá fronta může být způsobena velkým počtem *předmětů*, které jsou ve výrobní lince. Tím pádem to není problém ve výrobním procesu, ale spíše v plánování výroby. Místo toho se ukládájí data o *předmětu*, které pravděpodobně způsobiliy zpoždění při rychlém nárustu fronty v krátkém časovém intervalu. Fronty před každým strojem jsou průběžně kontrolovány a aktuální délka je porovnávána s délkou v minulosti, např. hodinu předtím. Pokud je rozdíl větší než hodnota zadaná uživatelem, poté se do databáze uloží produkty, které jsou právě zpracovávány ve stroji spolu s produkty, které ve stroji skončili do jedné hodiny od aktuálního času. Informace o produktech budou uloženy do relační tabulky a úkolem bude zjistit atributy *předmětu*, které se často vyskytují spolu v momentě, kdy fronta rychle narostla.

Po získání frekventovaných množin se provede filtrování. Pokud získáme velké množství množin s vysokou hodnotou *podpory*, je nutné porovnat tuto hodnotu s četností výskytu stejné frekventované množiny v celém záznamu událostí. Frekventovaná množina, která má stejnou nebo větší hodnotu podpory v celém záznamu událostí, nemá žádný význam pro analýzu zpoždění ve výrobním procesu. Hledají se takové frekvenentované množiny, které mají podporu významně větší při rychlém nárustu front, než v celém záznamu událostí. Z toho důvodu je potřeba zadávat minimální podporu nízkou, neboť hodně frekventovaných množin po dolování bude vyfiltrováno.

Toto filtrování nám umožní nalézt více zajímavé frekventované množiny. Je nejvíce pravděpodobné, že zpoždění na stroji je způsobeno nestandardními hodnotami atributů.

Tyto atributy se nevyskytují tak často v celé datové sadě, a proto je jejich podpora nízká. Avšak jejich informační hodnota pro náš případ dolování je vysoká.

Pro navrženou metodu filtrace byla definována nová hodnota, která je nazvána *procentuální změna podpory (PZP)* a je definována jako:

$$PZP(fm) = \frac{s_2(fm) - s_1(fm)}{s_1(fm)},$$

kde s_1 je hodnota podpory frekventované množiny fm v celém záznamu událostí a s_2 je hodnota podpory té samé množiny v datové sadě, která byla získána během rychlého nárustu front. Uživatel bude moci zadat minimální hodnotu PZP před samotným startem dolování.

Klasifikace na základě asociačních pravidel

V této dolovací metodě bude implementován postup, který byl popsán v Kapitole 4.3. Jelikož časy vykonání jednotlivých úkolů v záznamu událostí jsou numerické hodnoty, musí se pro klasifikaci diskretizovat na kategorické hodnoty. U diskretizace se musí provést dvě rozhodnutí. Za prvé zvolit diskretizační metoda a za druhé zvolit počet kategorií, do kterých se bude diskretizovat. První volba je blíže vysvětlena v Kapitole 6.4. Druhé rozhodnutí je již známo v návrhové fázi. Doba trvání byla diskretizována do tří kategorií, konkrétně na *krátké*, *střední* a *dlouhé* zpracování.

Většinou po trénovací fázi bude muset být proveden další krok, který vyfiltruje stejná asociační pravidla v různých kategoriích. Pro tento přístup jsou navrženy dvě metody:

- Stejná asociační pravidla v různých kategoriích jsou smazána.
- Asociační pravidlo s větší hodnotou podpory je ponecháno a je smázano to s menší hodnotou podpory.

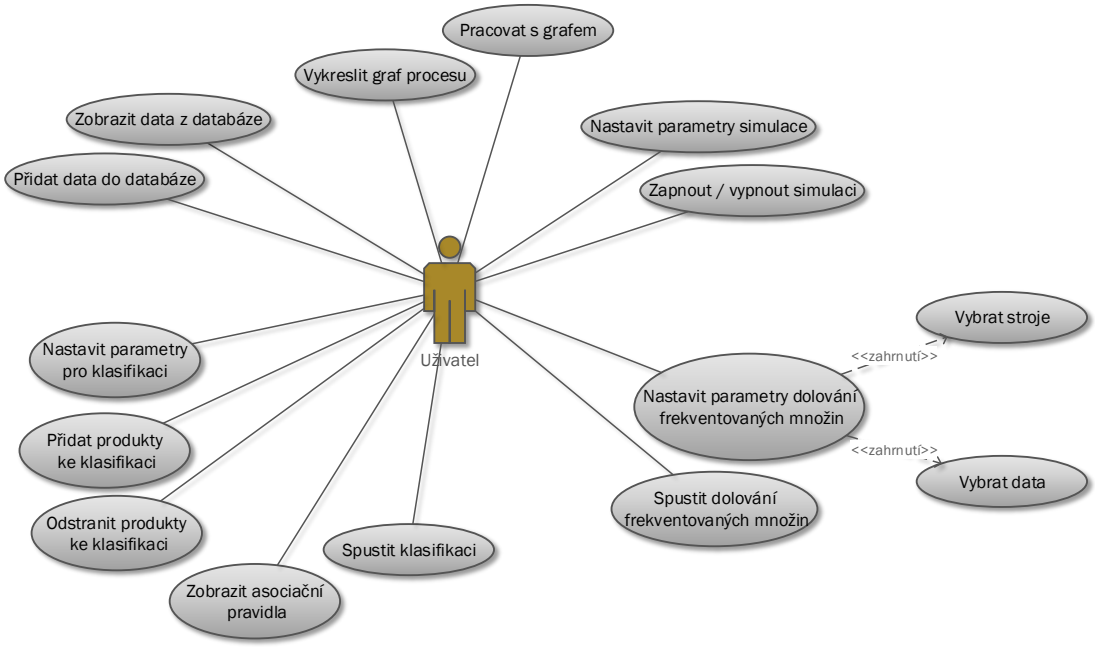
5.4 Specifikace požadavků

Cílem je navrhnout aplikaci, která bude umožňovat práci se zvolenými metodami pro dolování procesů. Hlavními kritérii při návrhu jsou srozumitelnost a jednoduché použití. Dalším prvkem při návrhu je dát volnost uživateli při výběru různých parametrů dolovacích metod.

Diagram případů užití navržené aplikace je znázorněn na Obrázku 5.1. V aplikaci se vyskytuje pouze uživatel, který s touto aplikací může pracovat. Je mu umožněno zobrazit si data z databáze, ve které je uložen celý záznam událostí a informace o jednotlivých produktech. Pokud je databáze prázdná, může uživatel nahrát data do databáze. Pokud jsou data v databázi, má uživatel možnost zvolit si jednu z implementovaných dolovacích metod a pracovat s ní.

U metody pro *objevení procesu* je možné pouze graf vykreslit a poté s ním různě interagovat. Graf je lehce interaktivní, po najetí myší na stroj jsou zvýrazněny cesty od uzlu a do uzlu. Dále je možné s jednotlivými uzly pohybovat po vykreslovací ploše.

U druhé dolovací metody (dolování frekventovaných množin při rychlém nárustu front), může uživatel pracovat se simulátorem a s ovládacími prvky samotného dolování. Tedy je mu umožněno nastavovat parametry simulace (krok simulace, datum začátku a konce simulace, rozdíl velikosti front při rychlém nárustu). Předtím než může uživatel spustit proces dolování, musí vybrat stroje, u kterých chce frekventované množiny zjišťovat a dále samotná



Obrázek 5.1: Diagram případů užití

data, ze kterých chce dolovat. Aplikace výsledky ze simulátoru ukládá do dynamicky vytvářejících se databázových tabulek. Uživatel tedy může vybrat i výsledky ze simulátoru, které byly do databáze vloženy někdy v minulosti. Uživatel může i smazat tabulky s daty ze simulátoru. Před spuštěním samotného procesu dolování se musí ještě nastavit hodnota minimální podpory a hodnota *PZP*.

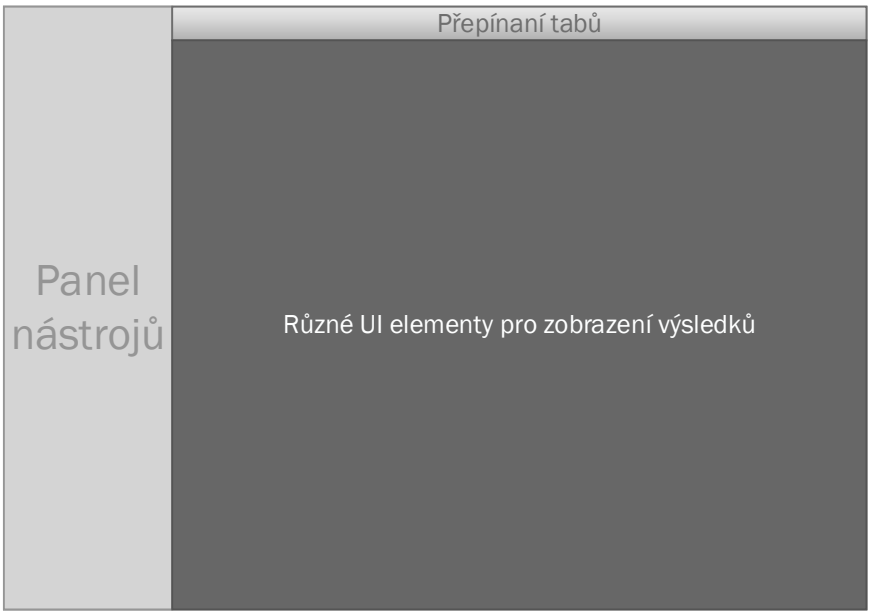
Požadavky pro třetí dolovací metodu (klasifikace na základě asociačních pravidel) jsou velmi podobné těm u druhé. Uživatel prvně musí vybrat stroj, u kterého se bude provádět klasifikace. Poté musí specifikovat produkty, u kterých se bude určovat klasifikační třída. Zde má tři možnosti. Buď si produkty specifikuje sám nebo si nechá vygenerovat úplně náhodné produkty, anebo si nahraje produkty z databáze, které někdy byly vybraným strojem zpracovány. Před startem klasifikační fáze uživatel ještě musí nastaví hodnotu minimální podpory a vybrat jednu ze dvou možností pro filtraci stejných asociačních pravidel. Jakmile vykonal všechny tyto úkony, může spustit klasifikaci. Trénovací fáze se provádí automaticky při změně nějakého parametru klasifikátoru, tudíž uživatel se tímto nemusí zabývat.

Výsledky dolovacích metod jsou zobrazeny v hlavním okně aplikace v tabulkové formě. Při běhu simulátoru jsou zobrazovány informace o strojích, konkrétně délka front a počet produktů, které dané stroje právě zpracovávají. Aplikace také umožňuje zobrazení asociačních pravidel pro jednotlivé klasifikační kategorie u všech specifikovaných produktů a také u vybraného stroje.

5.5 Grafický návrh

Všechny grafické návrhy v této kapitole mají znázorňovat základní grafickou koncepci aplikace a nepředstavují finální vzhled, který se v průběhu implementační fáze mohl změnit.

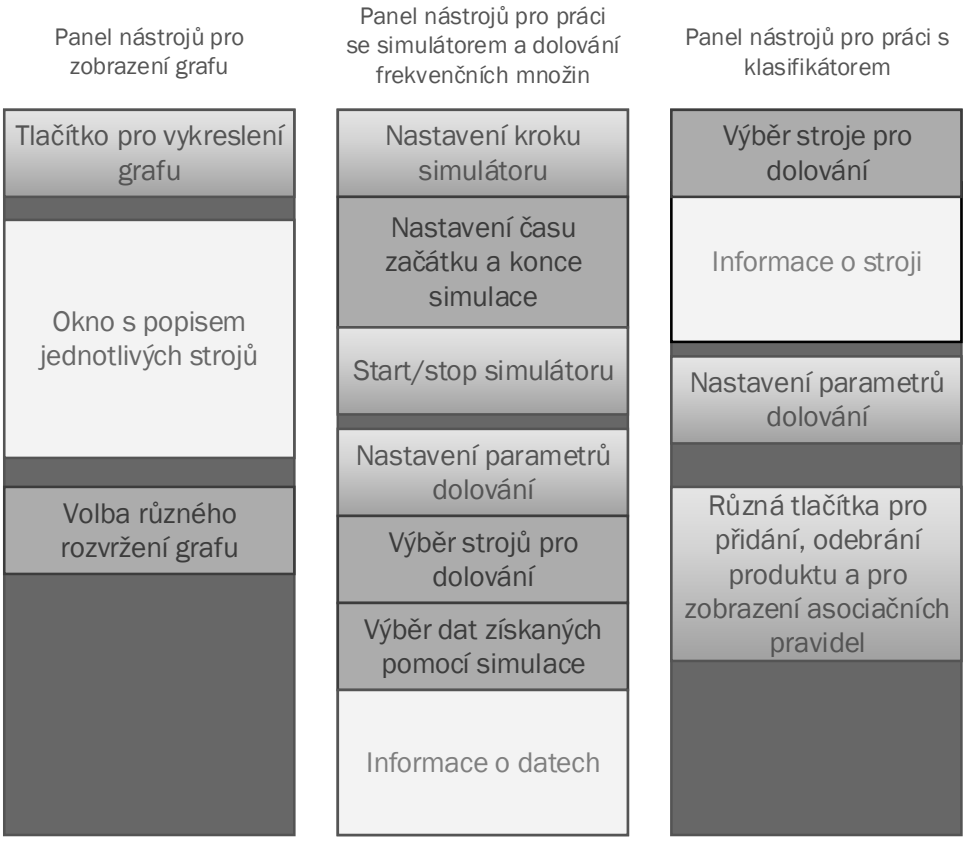
V předešlých kapitolách byl popsán princip dolovacích metod. Cílem návrhu grafického uživatelského rozhraní je co nejvíce zjednodušit práci s těmito metodami a umožnit uživateli rychle a efektivně měnit různé parametry pro dolování znalostí. Dále je cílem co nejmenší počet vyskakovacích oken, tudíž skoro všechny ovládácí prvky jsou zahrnuty v hlavním okně aplikace. Pouze zobrazení získaných asociačních pravidel bude mít své specíální okno. Návrh grafického uživatelského rozhraní hlavního okna aplikace je zobrazen na Obrázku 5.2.



Obrázek 5.2: Grafiký návrh hlavního okna aplikace

Vzhled výsledné aplikace bude implementována v grafickém stylu posledního operačního systému Windows 8 od společnosti Microsoft, který je nazývan *Modern UI*. K základním aspektům tohoto uživatelského rozhraní patří jednoduchost, více zaměření na obsah a typografii.

Na Obrázku 5.3 jsou vidět návrhy tří různých panelů nástrojů pro tři různé metody dolování. Tyto panely budou výhradně sloužit pro nastavování parametrů metod pro získávání znalostí. Budou se přepínat podle výběru dolovacích metod. Dále zde budou zobrazeny některé informace o zvolených strojích nebo o zvolených datech získaných ze simulátoru produkční historie.



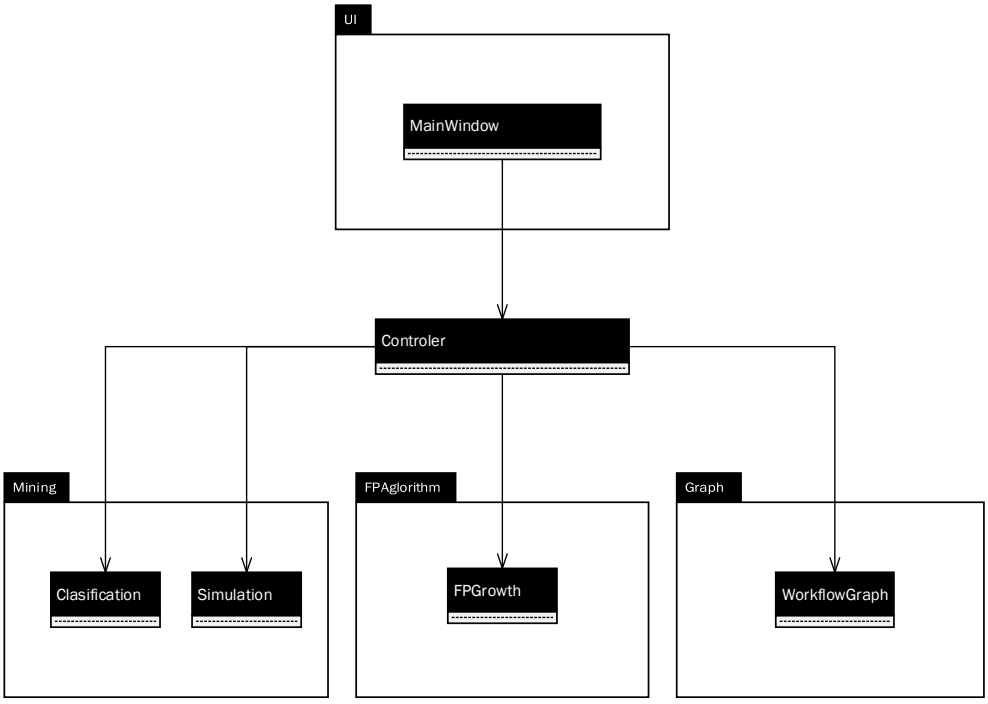
Obrázek 5.3: Grafiký návrh tří panelů nástrojů

Kapitola 6

Implementace navržené aplikace

Obsahem této kapitoly je popis implementace navržené aplikace pro získávání znalostí z výrobního procesu. Aplikace je implementována v programovacím jazyce *C# .NET verze 4.5*. Grafické uživatelské rozhraní je vytvořeno pomocí *Windows Presentation Foundation* (WPF), což je podmnožina *.NET* pro tvorbu *RUI* (Rich User Interface). Data poskytnutá k dolování jsou ve formátu `.csv`. Aby se s nimi dalo lépe pracovat, byla vytvořena lokální service-based databáze SQL. Z toho důvodu je nutné mít na počítači, kde se aplikace spouští, nainstalovaný SQL Server Express 2012 od společnosti Microsoft. Výsledný vzhled aplikace s popisem použití je na Obrázku A.1 v příloze.

Na Obrázku 6.1 je zobrazen zjednodušený diagram tříd, který ukazuje hlavní vztahy v celé aplikaci. Detailnější diagramy tříd pro všechny balíčky budou ukázány dále v kapitole a podrobně popsány.



Obrázek 6.1: Zjednodušený diagram tříd výsledné aplikace

6.1 Algoritmus Frequent Pattern Growth

Následující odstavce obsahují popis tříd, kterými je implementován algoritmus *Frequent Pattern Growth* popsaný v podkapitole 4.2, a jehož diagram tříd je znázorněn na Obrázku 6.2.

Třída `FPGrowth`

Algoritmus pro získání frekventovaných množin *FP Growth* je implementován v třídě `FPGrowth` a spouští se zavoláním jediné `public` metody `RunAlgorithm`. Vstupními parametry této metody jsou seznam řetězců (transakce), hodnota minimální podpory a příznak, který určuje typ formátu vydolovaných frekventovaných množin. Práce algoritmu se dá rozdělit do několika kroků:

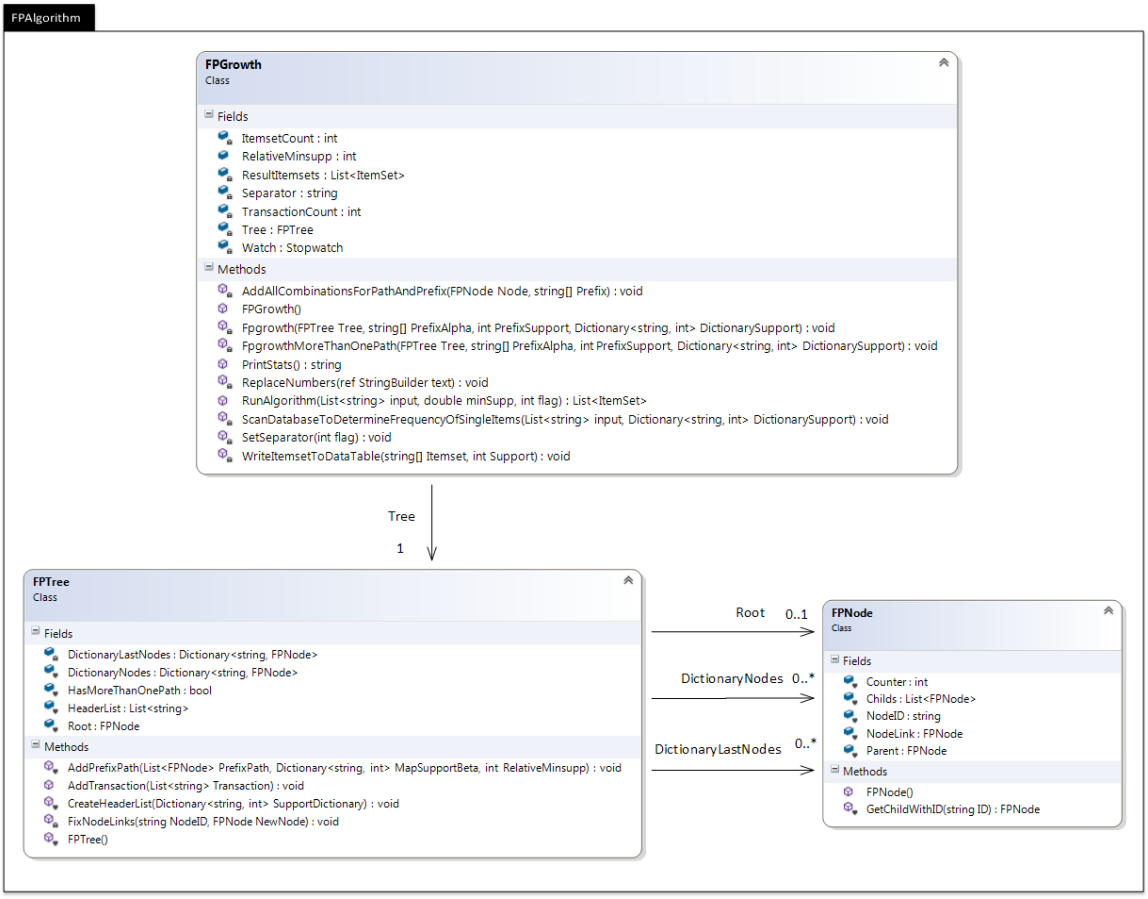
- Prvním krokem je skenování databáze pomocí metody `ScanDatabaseToDetermineFrequencyOfSingleItems`, která rozdělí vstupní transakce na jednotlivé prvky. Pro každý prvek určí hodnotu podpory v celém vstupním seznamu transakcí.
- V druhém kroku se znovu prochází vstupní seznam transakcí a vytváří se *FP strom*. Před vložením transakce do stromu jsou jednotlivé prvky transakce seřazeny podle hodnoty podpory. V případě, že jsou hodnoty podpory stejné, jsou prvky seřazeny podle abecedy.
- Následně se vytvoří seznam hlaviček výsledného stromu.
- Poslední krok již představuje samotné dolování frekventovaných množin. Dolování je prováděno rekursivním voláním metody `FPgrowth`. Tato metoda zjistí, jestli *FP strom* obsahuje jednu cestu nebo více cest. V prvním případě se volá metoda `AddAllCombinationsForPathAndPrefix`, která vytváří všechny možné kombinace frekventonavaných množin z dané cesty, které mají hodnotu podpory větší, než je minimální podpora. V případě, že strom obsahuje více jak jednu cestu, se volá metoda `FpgrowthMoreThanOnePath`. Tato metoda vytváří subdatabázi (seznam), která představuje množinu prefixových cest ve stromě, které se společně vyskytují se *suffix* vzorem. Následně se vypočíta podpora všech prefixových cest v získané subdatabázi a zkonstuuje se FP strom, který je tvořen všemi prefixovými cestami v subdatabázi. Tento strom se poté rekurzivně doluje pomocí funkce `FPgrowth`.

Algoritmus v průběhu dolování volá metodu `WriteItemsetToDataTable`, která vytváří výslednou podobu frekventovaných množin ve formě instance třídy `ItemSet`. Instance je přidána do seznamu, který je na konci dolování vracen jako návratová hodnota. V případě potřeby je také možno zavolat metodu `PrintStats`, která tiskne statistiky o posledním dolování. Jedná se o počet vstupních transakcí, počet vydolovaných frekventovaných množin a čas dolování.

Třídy `FPTree` a `FPNode`

Instance třídy `FPNode` představují uzly v *FP stromě*. Každý uzel je reprezentován svým ID. Dále obsahuje hodnotu podpory, odkaz na rodiče v *FP stromě*, seznam svých potomků a odkaz na další uzel se stejným jménem. Jediná metoda, kterou instance třídy nabízí, je `GetChildWithID`, která vrací odkaz na uzel s hledaným ID nebo `null`.

Instance třídy `FPTree` představuje *FP strom*. Tento strom v sobě obsahuje odkaz na kořen, hash mapu jednotlivých uzlů, seznam hlaviček a hash mapu listů. Do stromu je možné přidávat uzly pomocí dvou různých metod. První metoda `AddTransaction` se volá při konstrukci původního *FP stromu* v rámci skenování databáze. Vytváří jednotlivé uzly ve stromě a propojuje je mezi sebou. Druhá metoda `AddPrefixPath` se volá při konstukci prefixového stromu ze subdatabáze. Pracuje podobně jako první metoda, s tím rozdílem, že se musí procházet všechny uzly z prefixové cesty a z nich se vytváří jednotlivé uzly *FP stromu*.



Obrázek 6.2: Diagram tříd balíčku FPAlgorithm.

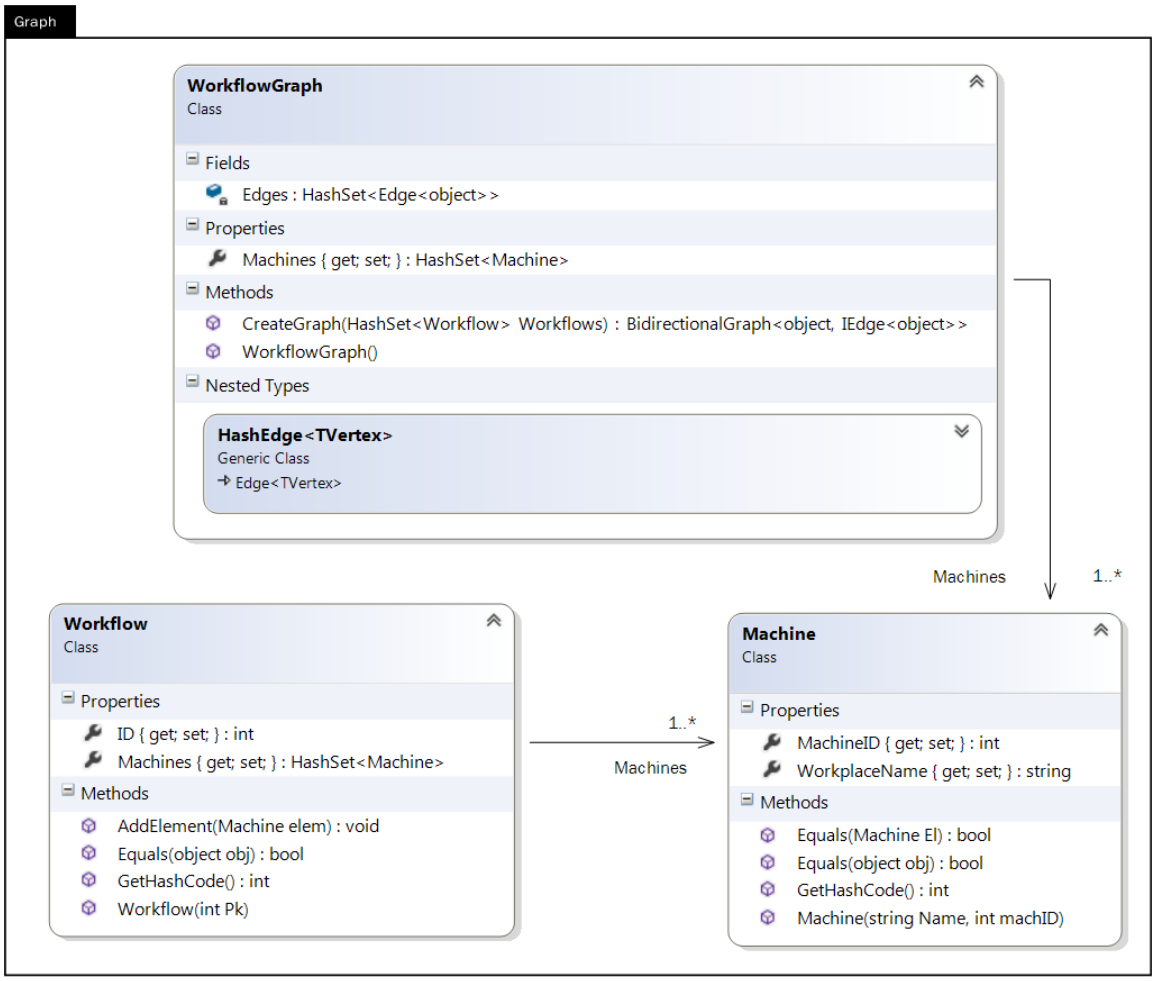
6.2 Dolování grafu výrobního procesu

Pro reprezentaci grafu procesu se využívají tři třídy: `Machine, Workflow a WorkflowGraph`. Třída `Machine` představuje jeden stroj ve výrobním procesu a každá instance této třídy je identifikována pomocí ID, které je stejné jako ID stroje ve vstupních datech. Třída nenabízí žádné metody, pouze přepisuje metody `Equals a GetHashCode`, neboť se instance této třídy používá v kolekci `HashSet`.

Třída `Workflow` představuje jednu cestu produktu výrobním procesem. Obsahuje kolekci `HashSet`, ve které jsou obsaženy všechny stroje, na kterých byl jeden daný produkt zpracován. Opět jako v případě třídy `Machine` jsou přepsány metody `Equals a GetHashCode`,

neboť i instance této třídy se budou vkládat do kolekce `HashSet`. Pro vykreslení grafu totiž chceme mít všechny možné cesty procesem, ale každá cesta nám stačí pouze jedenkrát.

Třída `WorkflowGraph` reprezentuje graf výrobního procesu. Graf se skládá z uzlů a hran mezi uzly. Proto tato třída obsahuje kolekci strojů, které představují jednotlivé uzly grafu. A také obsahuje kolekci propojení mezi stroji, což jsou hrany. Tato třída používá knihovnu `QuickGraph`, která poskytuje rozhraní pro reprezentaci různých grafů. V našem případě bylo použito rozhraní `BidirectionalGraph`, pomocí kterého se implementuje orientovaný graf. Graf se sestrojuje pomocí jedné jediné metody `CreateGraph`, která jako parametr potřebuje `HashSet` všech možných cest grafem v podobě instancí třídy `Workflow`. Metoda tuto kolekci prochází a postupně vytváří všechny uzly a hrany grafu. Jakmile je proces dolování hotov, metoda vrátí instanci grafu, která je poté vykreslena v hlavním okně aplikace. Pro vykreslení grafu se používá knihovna `GraphSharp`. Graf je mírně interaktivní. Je možné pohybovat s jednotlivými uzly, přibližovat a oddalovat pohled na graf a po najetí myší na uzel se zvýrazní hrany, které směřují do uzlu i ven z uzlu. Diagram tříd balíčku `Graph` je zobrazen na Obrázku 6.3.



Obrázek 6.3: Digram tříd balíčku Graph.

6.3 Simulátor produkční historie

Třída Event

Jelikož se dolování provádí ze záznamu událostí, bylo nejprve nutné vytvořit třídu pro jeden záznam. Každý záznam je jedinečně identifikovatelný čtyřmi atributy:

- `WorkplaceName` - id pracoviště, na kterém je daná událost spuštěna.
- `Start` – čas začátku události.
- `End` – čas konce události.
- `ProductID` – jednoznačný identifikátor produktu, který se na daním stroji zpracovává.

Žádne dvě události v záznamu nemohou mít všechny tyto atributy stejné.

Třídy `ProcessingProduct, WorkplaceSimulation a FrontOfWorkplace`

Tyto tři třídy abstrahují reálný výrobní proces. Instance třídy `ProcessingProdukt` představují produkty, které se aktuálně nacházejí ve výrobním procesu. Jsou dynamicky vytvářeny a rušeny podle aktuálního času simulátoru. Každý produkt je dán svým ID, které se získává ze záznamu událostí a dále obsahuje seznam všech událostí, které mají být vykonány. Tyto události jsou velice důležité, neboť podle nich se produkt v simulátoru řídí.

Třída `WorplaceSimulation` slouží pro reprezentaci pracoviště v procesu. Jedno pracoviště obsahuje tři seznamy. V prvním seznamu `ProcessingEvents` jsou umístěny právě zpracovávané produkty. Druhý seznam `Front` představuje frontu před pracovištěm. A poslední seznam `OneHourOldEvents` obsahuje události, které na pracovišti skončili před méně než jednou hodinou. Tento seznam se využívá při vkládání produktů do datové sady po rychlém nárustu fronty, neboť nás zajímají nejen produkty aktuálně zpracovávané strojem, ale i produkty, které se ve stroji nacházeli maximálně před hodinou a mohly tak způsobit zpomalení výrobního procesu.

Třída `FrontOfWorkplace` je jednoduchá třída, která obsahuje jméno pracoviště a délku fronty. Třída sama v sobě nemá časový údaj, kdy byla délka fronty aktuální. Instance této třídy se vkládají do speciální kolekce `SortedDictionaryExtensions` jako seznam všech pracovišť. Kolekce je rozšířením `SortedDictionary`, kde klíč je hodnota času simulátoru při vložení a obsah je seznam všech pracovišť jako instance třídy `FrontOfWorkplace`. Fronty se do kolekce vkládájí po jedné hodině, ale krok simulace může být libovolný. Tím pádem mohou vzniknout klíče, které by nebylo možno vyhledat. Řešením je právě tato kolekce, která dokáže najít nejbližší klíč k zadanému.

Třída `Simulation`

V třídě `Simulation` je implementován simulátor produkční historie. Před samotným spuštěním simulátoru je nutné přidat data do instance této třídy. Data se přidávájí automaticky, uživatel pouze zvolí začátek a konec simulace v hlavním okně aplikace. Algoritmus si poté sám stáhne data z databáze podle určených časů. Jakmile má simulátor data k dispozici, zavolá se ještě metoda `PrepareData`, která zjistí všechny stroje v záznamu událostí a připraví data pro simulaci. Samotný simulátor pracuje v krocích, které uživatel zadává z hlavního okna. Délka kroku může být libovolná. Pokud je nastavená hodnota větší jak

3600 (1 hodina), pak simulátor používá hodnotu kroku pouze na výpisy a vnitřně pracuje po hodinách.

Simulátor vykonává svoji činnost pomocí metody `SimulationStep` s parametrem délky kroku simulace. Tato metoda se dá popsat v několika bodech:

- Na začátku se pomocí metody `CheckProcessingProduct` zjišťuje, zda-li v simulátoru není nějaký zapomenutý produkt, který má menší čas poslední události než je aktuální čas simulace.
- V druhém kroku se porovnáva aktuální čas simulace s nastaveným koncovým stavem. Touto podmínkou se ukončuje běh simulátoru.
- V dalším kroku se již provádí samotná simulace tím, že se prochází záznam událostí a pro každou událost se může vykonat několik metod. Záznam se prochází jen po události, které mají časy menší než je aktuální čas + krok simulace.

  V rámci tohoto kroku se vezme událost pomocí metody `GetEvent` a podle toho, jestli událost právě končí nebo začíná, se vykonají další metody. Při startu události se do stroje přidá zpracovávaná událost. Dále se vytvoří produkt jako instance třídy `ProcessingProduct`, pokud v kolekci `ProcessingProducts` zatím neexistuje. Poslední volaná metoda je `RemoveProductFromFront`, která odstraní produkt z fronty před strojem. Při konci události se vykonávají jiné metody. Nejprve je odstraněna událost z pracoviště a ta stejná událost je přidána do hodinu starých událostí v pracovišti. Dále se pro produkt zjistí příští pracoviště, do jehož fronty je produkt přidán (metoda `AddProductToFront`). V případě, že produkt již nemá další naplánovanou událost, je instance třídy `ProcessingProduct` odstraněna ze simulátoru.
- Posledním krokem je porovnání délky aktuálních front s délkami front před hodinou. Pokud je u některého pracoviště zjištěn příliš velký nárust fronty, jsou do databáze uloženy právě zpracovávané produkty v pracovišti spolu s produkty, které skončili zpracování méně než před hodinou.

Po každém kroku simulátoru je v hlavním okně vypsán aktuální počet produktů a délka fronty pro všechny pracoviště. Výpis je naformátován tak, že pokud došlo ke změně nějakého údaje oproti minulému výpisu, tak je tento údaj zvýrazněn červeně. Aktuální stav simulátoru je možno získat pomocí metody `GetSimulationState`. Celý běh simulace je zpracováván pomocí `BackgroundWorkeru`, tzn. že simulace běží v jiném vlákně než je spuštěno grafické uživatelské rozhraní. Tím je umožněno pracovat s aplikací a zároveň mít spuštěný simulátor.

Produkty získané ze simulátoru jsou ukládány do databáze do dynamicky se vytvářející tabulky. Nová tabulka, do které se budou ukládat data, je vytvořena vždy po resetu simulátoru.

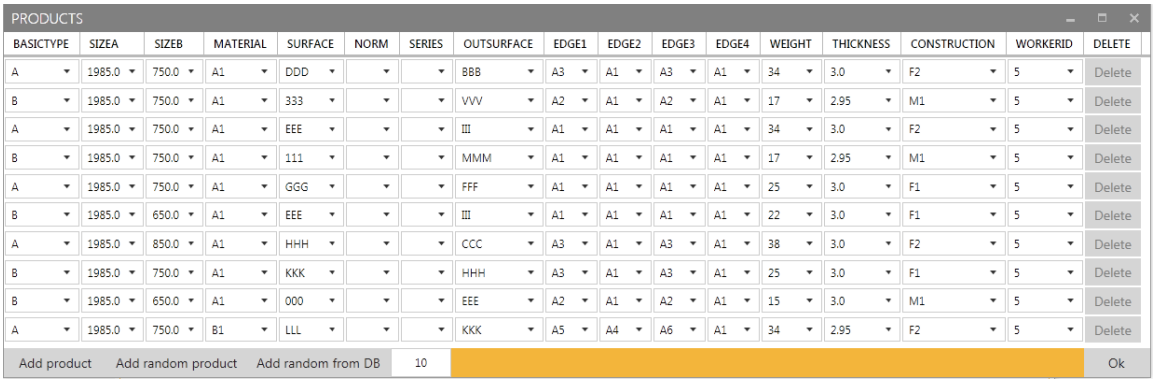
6.4 Klasifikátor

Klasifikátor je implementován ve třídě `Clasification`. Klasifikace pracuje ve dvou fázích. První fáze je učení klasifikátoru. Druhá fáze je již samotná klasifikace. Aplikace sama pozná, kdy je potřeba spustit trénovací fázi. Jakákoliv změna parametru klasifikátoru zapříčíní nastavení příznaku, který klasifikátoru řekne, že před samotnou klasifikací má spustit trénovací fázi.

Při startu aplikace se musí klasifikátor inicializovat. To znamená zjistit všechny pracoviště ze záznamu událostí a pro každé pracoviště zjistit časové intervaly, podle kterých se bude klasifikovat do kategorií. K tomu slouží dvě metody `GetAllWorkplaces` a `GetExecutionTimes`. Druhá zmíněná metoda je velice důležitá a pracuje následovně. Pro každé pracoviště se zjistí všechny časy zpracování produktů, které prošly daným pracovištěm. Časy se seřadí vzestupně od nejmenšího po největší a uloží se do seznamu. V našem případě hledáme dva konkrétní časy, které by seznam rozdělili do kategorií. To je provedeno tak, že se zjistí délka seznamu a vydělí se třema. Tuto hodnotu nazveme např. *index*. Poté kategorie reprezentující *krátký čas* je dána intervalem od nuly po čas, který je v seznamu na pozici `index`. Kategorii *střední čas* dostaneme podobně. Spodní hranice intervalu je čas, který je větší než čas na pozici `index` a horní hranice invervalu je dána časem, který je na pozici `index * 2`. Poslední kategorie *dlouhý čas* je poté dána jako interval, kde spodní hranice je větší než čas na pozici `index * 2` a horní hranice není specifikována.

Trénovací fáze spočívá v získání asociačních pravidel pro všechny tři kategorie pomocí `FP algoritmu`. V aplikaci stačí nastavit minimální podporu a vybrat jednu ze dvou možností pro filtraci asociačních pravidel. Při výbrání první možnosti (Unique association rules) se odstraní pravidla, která nejsou unikátní. To znamená, že pokud se ve dvou různých kategoriích nachází stejné pravidlo, tak se u obou kategorií odstraní. Při druhé možnosti (Higher support) se porovnává podpora pravidel, které nejsou unikátní. Odstraní se to pravidlo, které má hodnotu podpory menší.

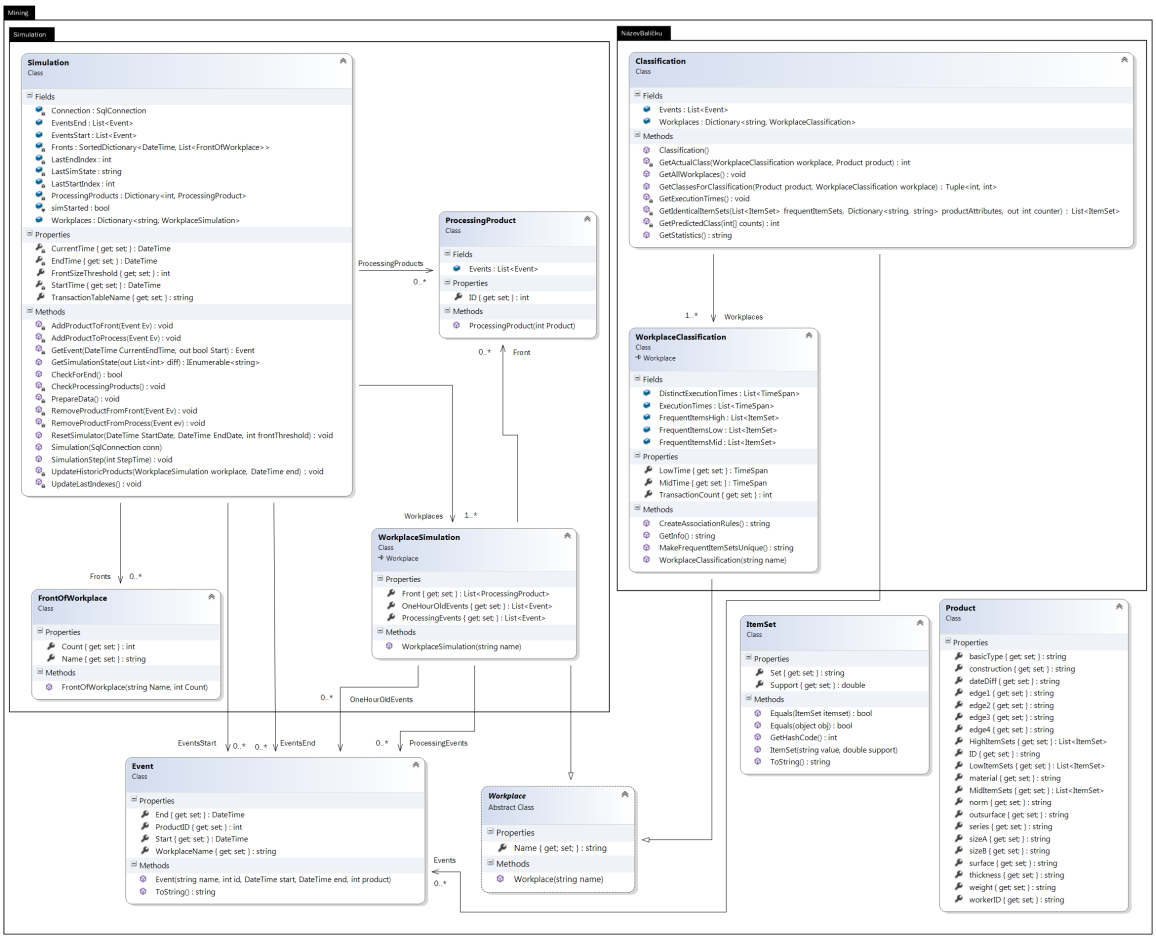
Po trénovací fázi se pokračuje testovací/klasifikační fází. Nejprve je nutné v aplikaci přidat produkty ke klasifikaci. Okno s přidáváním produktů je vidět na Obrázku 6.4



Obrázek 6.4: Ukázka okna s produkty.

Jakmile jsou produkty přidány, spustí se podobně jako u simulace `BackgroundWorker`, který začne klasifikovat na pozadí v jiném vlákně. Pro tuto fázi je ve třídě `Clasification` k dispozici metoda `GetClassificationClasses`. Tato metoda porovná atributy zadaného produktu s asociačními pravidly ve všech kategoriích. Delší asociační pravidla jsou lépe ohodnocena než kratší. Jakmile jsou známy jednotlivá ohodnocení produktu pro všechny kategorie, přiřadí se mu klasifikační třída a dále se zjistí skutečná třída produktu. Do hlavního okna aplikace se vypíše výsledek klasifikace ve formě kontingenční tabulky.

Balíček `Mining`, ve kterém jsou umístěny třídy pro klasifikaci a simulaci, je zobrazen na Obrázku 6.5.



Obrázek 6.5: Diagram tříd balíčku Mining.

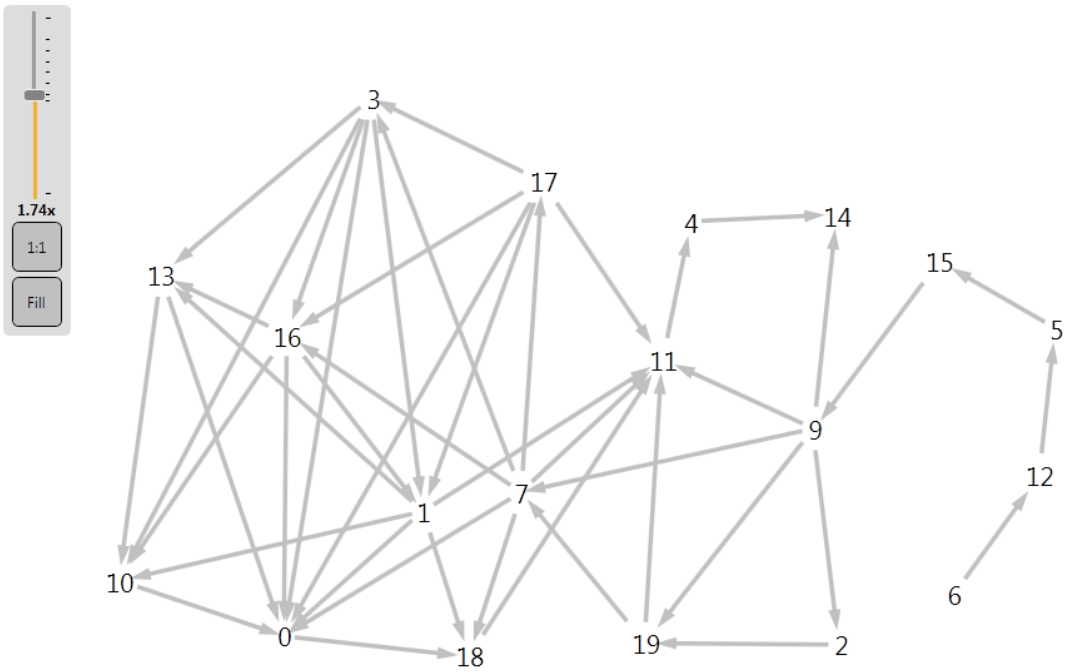
Kapitola 7

Výsledky dolování

V následující kapitole budou popsány a diskutovány některé experimenty, které byly prováděny s implementovanými algoritmy.

7.1 Výsledný graf výrobního procesu

S touto metodou se nedají provádět žádné experimenty. Cílem je pouze vykreslit propojení mezi jednotlivými stroji. V aplikaci se dá u grafu měnit rozložení jednotlivých uzlů, ale to na výsledky nemá žádný vliv. Výsledný graf je vidět na Obrázku 7.1. Bohužel tento graf není moc dobře pochopitelný z obrázku, neboť jsou zobrazeny pouze ID strojů tak, jak jsou uloženy v poskytnutých datech.



Obrázek 7.1: Výsledný graf výrobního procesu.

7.2 Dolování frekventovaných množin

Záznam událostí, se kterým se prováděla simulace produkční historie, obsahuje přibližně 20000 záznamů a každý produkt je popsán 28 atributy. V našem případě bylo k dolování vybráno pouze 16 nejdůležitějších atributů, které popisují vlastnosti produktu. Některé atributy představují dobu expirace nebo různé příznaky, které jsou skoro u všech produktů stejné, tudíž by na výsledky neměly žádný vliv.

Při nastavení simulátoru tak, aby prošel celý záznam událostí a ukládal produkty jen při nárustu fronty o více jak 100 produktů za hodinu, dostaneme u různých strojů různé počty produktů. Výsledek je vidět na Obrázku 7.2.

Workplace	Cnt
machine 2.1	162
line X.1	86
line X.2	42
line X.6	31
shell prep.	22
machine 1.1	8
press	100

Obrázek 7.2: Výsledné počty produktů po simulaci.

Experimenty u této metody spočívaly v nastavování parametru minimální podpory a hodnoty *PZP*. V aplikaci je možné vybrat pracoviště, u kterého chceme dolovat frekventované množiny. Aplikace dovoluje vybrat i více pracovišť naráz, avšak výsledky mohou být jiné než pro jedno pracoviště, neboť se zvýší počet transakcí a tím se změní i podpory jednotlivých atributů. Bylo provedeno několik experimentů s různými nastaveními minimální podpory a hodnoty *PZP*. Nejzajímavější výsledky jsou vidět v tabulkách níže. V první Tabulce 7.1 jsou vidět různá nastavení pro jedno vybrané pracoviště. V druhé tabulce 7.2 jsou vidět různá nastavení pro více vybraných pracovišť.

Min. podpora	**Po FP Growth**	**PZP = 0,10**	**PZP = 0,15**	**PZP = 0,20**	**PZP = 0,25**
0,10	16835	980	829	726	677
0,15	8719	582	437	346	313
0,20	3693	202	163	107	83
0,25	2205	96	65	54	36
0,30	1099	84	53	44	26

Tabulka 7.1: Tabulka výsledků při různých nastavení s jedním pracovištěm.

Min. podpora	**Po FP Growth**	**PZP = 0,10**	**PZP = 0,15**	**PZP = 0,20**	**PZP = 0,25**
0,10	7797	1061	954	882	677
0,15	3517	532	437	398	341
0,20	2033	190	156	132	98
0,25	898	75	53	38	28
0,30	616	58	39	26	20

Tabulka 7.2: Tabulka výsledků při různých nastavení s třemi pracovišt.i

Cílem této metody dolování je získat dostatek frekvenčních množin, které nejspíše způsobily zpomalení výrobního procesu. Musíme tedy najít optimální hodnoty minimální podpory a *PZP*. Z tabulek vyplývá, že je ideální nastavit minimální podporu na hodnoty od 0,15 do 0,25. Z tabulek dále vyplývá, že hodnota *PZP* má silný vliv na počet získaných frekventovaných množin. Při vyšším nastavení tohoto parametru je mnoho množin vyfiltrováno. Optimální hodnota pro získání dostatečného počtu množin je mezi hodnotami 0,15 - 0,20.

Na délku získaných frekventovaných množin nemají nastavení moc velký vliv. Obvykle délka nepřesáhne hodnotu 6. Průměrně však dosahuje hodnoty 4. Nejčastější atributy ve frekventovaných množinách se týkají různých vizuálních parametrů předmětu. Jen málo množin obsahuje informace o rozměrech nebo jiné numerické hodnoty.

Z experimentů bylo zjištěno, že zpomalení výrobního procesu je způsobeno hlavně specifickými hodnotami kategorických atributů, které určují vizuální podobu předmětu.

7.3 Klasifikace na základě asociačních pravidel

Experimenty u této dolovací metody zahrnovali nastavování minimální podpory a různé filtrování stejných asociačních pravidel. Hned na začátek musím konstatovat, že úspěšnost klasifikace byla od 40% do 70% pro různá pracoviště. Dále si shrneme některé možné příčiny takto malé úspěšnosti. Nejprve si však ukážeme výsledek jedné klasifikace v podobě kontingenčních tabulky z aplikace.

Na prvním Obrázku 7.3 je vidět výsledek trénovací fáze u jednoho pracoviště s minimální podporou 0,25 a *unikátním* filtrovacím pravidlem. Jsou zde zobrazeny výsledky z algoritmu *FP Growth* a počty asociačních pravidel před filtrací a po ní. Dále můžeme vidět čas potřebný k dolování frekventovaných množin, který je opravdu nízký, jak bylo naznačeno v teorii o tomto algoritmu.

```
================================================================================
========= Settings of classification has changed - new training phase ===========
=============  FP-GROWTH - STATS ==================
 Transactions count from database : 7640
 Frequent itemsets count : 1087
 Total time ~ 00:00:00.1266894
================================================
=============  FP-GROWTH - STATS ==================
 Transactions count from database : 6259
 Frequent itemsets count : 757
 Total time ~ 00:00:00.0955519
================================================
=============  FP-GROWTH - STATS ==================
 Transactions count from database : 4750
 Frequent itemsets count : 669
 Total time ~ 00:00:00.1315749
================================================
Itemsets before prunning:
1087  757  669
Itemsets after prunning:
571  148  318
================================================================================
```

Obrázek 7.3: Výsledek trénovací fáze.

Výsledek klasifikace pro 10000 produktů je zobrazen v Tabulce 7.3. Je vidět, že prostřední kategorie nebyla skoro nikdy klasifikována, což je největší nedostatek celé klasifikace.

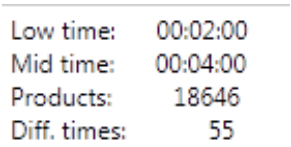
Ukázka tohoto výsledku byla vybrána záměrně, neboť vystihuje dva problémy, o kterých si myslím, že přispívají k horší klasifikaci. Ostatní experimenty se chovaly podobně s menšími odchylkami.

		Predikovaná třída		
	Čas	Nízký	Střední	Vysoký
Realná třída	Nízký	2302	0	1745
	Střední	1669	1	1721
	Vysoký	1023	1	1527

Tabulka 7.3: Kontingenční tabulka klasifikační fáze.

1. Z Obrázku 7.3 můžeme vidět výsledné počty asociačních pravidel pro všechny tři kategorie. Již tu vzniká problém, neboť první kategorie obsahuje dvakrat více pravidel než třetí a asi 4 krát více než kategorie druhá. Z toho plyne, že pokud se budou klasifikovat produkty, které naleznou shodu ve všech kategoriích, je nejvíce pravděpodobné, že u první kategorie se těchto pravidel nalezne nejvíce. Změna druhu filtrace (zachování pravidla s větší hodnotou podpory) tomuto problému občas pomůže, občas ne. Záleží na vybraném stroji. Může se stát, že poměry se ještě zvětší kvůli vyšší podpoře asociačních pravidel v první kategorii. Ani změna hodnoty minimální podpory nepomáhá, neboť poměr počtu asociačních pravidel zůstává stejný.

2. Pro nastínění druhého nedostatku použijeme Obrázek 7.4 z aplikace, který ukazuje informace o rozdělení časových intervalů do košů. Je potřeba zmínit, že poskytnutá data obsahují časy zaokrouhlené na celé minuty. První kategorie představuje interval od 0 po 2 minuty. Druhá kategorie obsahuje pouze časy zpracování 3 a 4 minuty. Poslední kategorie od 5 minut výš. Parametr *Diff. times* ukazuje, kolik bylo různých časů zpracování na pracovišti. Vidíme, že 3 časy jsou v první kategorii, pouze 2 časy v druhé a zbytek v poslední. Pro lepší výsledky klasifikace by bylo potřeba lépe určit intervaly časů zpracování. Bohužel já nemám k dispozici potřebné informace, které časy zpracování jsou nízké, normální nebo vysoké. V tomto případě by bylo potřeba experta z výrobní továrný, který by lépe intervaly určil.



Obrázek 7.4: Ukázka informací o časech zpracování.

Metoda pro klasifikaci na základě asociačnch pravidel se jeví pro náš výrobní proces jako méně použitelná. Avšak domnívám se, že po menší úpravě intervalů jednotlivých kategoriíí by bylo možno dosáhnout mnohem lepších výsledků, pokud by se do návrhu přidali znalosti experta na daný výrobní proces. Zjistil jsem však, že výrobní časy produktů nejsou závislé pouze na atributech, ale i na některých vnějších faktorech, neboť získaná asociační pravidla byla velmi podobná, ne-li stejná, ve všech kategoriích.

Kapitola 8

Závěr

Dolování procesů je důležitým nástrojem moderních organizací, které potřebují řídit netriviální operační procesy. Na jedné straně je enormní nárust zaznamenávaných dat. Na druhé straně procesy a informace musí být přesné, aby splňovaly požadavky spojené s efektivností, rychlostí a úspěšností. Digitální a fyzikální svět se slučuje do jednoho světa, kde události jsou zaznamenávány v čase vykonání a procesy jsou usměrňovány a kontrolovány na základě datových záznamů.

Cílem práce bylo nastudování získávání znalostí z obchodních procesů a implementace zvolených metod dolování. Na základě návrhu zvolených metod byla implementována aplikace v programovacím jayzce C#.

Práce shrnuje poznatky o získávání znalostí z obchodních procesů. Nejprve je vysvětleno obecné získávání znalostí z databází, na které je návázáno rozborem právě problematiky dolování procesů. Je ukázána role dolování procesu v životním cyklu procesního řízení a různé typy dolování. Značná část se věnuje objevení procesu, neboť tento typ dolování je považován za nejvíce náročný. Další kapitola práce se věnuje dolovacím technikám, které byly použity v návrhové a implementační části.

Na základě analýzy vybraného výrobního procesu byly navrhnuty tři typy dolovacích metod, jejichž výsledky by napomohly lepšímu pochopení výrobního procesu a zefektivnily by samotnou výrobu. První metoda je objevení procesu, která zobrazuje graf propojení jednotlivých strojů v procesu. Druhá metoda využívá simulátor produkční historie, který zjišťuje rychlý nárust front před pracovišti a ukládá informace o produktech, které pravděpodobně způsobily zpomalení výrobního procesu. Následně se dolují frekventované množiny z dat získaných pomocí simulátoru. Výsledky nám udávají, které množiny atributů nejspíše způsobují opoždění na strojích a na základě těchto výsledků je možno lépe plánovat výrobní proces. Poslední navržená metoda je klasifikace na základě asociačních pravidel, která má za cíl určit přibližnou dobu zpracování produktu na výrobním pracovišti. K tomu se využívá klasifikace do tří tříd. Výsledky metody by se daly použít pro predikci doby zpracování produktu a tím zajistit lepší plánování nebo odhad produktů, které by mohly zapříčínit zpomalení nebo dokonce zastavení procesu výroby.

Experimenty ukázaly, že druhá navržená metoda je plně funkční a dá se pomocí ní zjistit frekventované množiny, které způsobily zpomalení procesu výroby. Velký vliv na počet vydolovaných množin má parametr *PZP*. Třetí metoda se ukázala jako méně funkční. Výsledky klasifikátoru nesplňují očekávání. Bylo by potřeba lepší rozdělení intervalů časů zpracování. Informace o intervalech by musel dodat nějaký expert z výrobní továrny.

Literatura

[1] van der Aalst, W. M. P.: *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011, ISBN 978-3-642-19344-6.

[2] van der Aalst, W. M. P.; Reijersa, H. A.; Weijtersa, A. J. M. M.; aj.: Business process mining: An industrial application. In *Information Systems*, ročník 32, 2007, ISSN 0306-4379, s. 713–732.

[3] van der Aalst, W. M. P.; Weijters, T.; Maruster, L.: Workflow mining: Discovering process models from event logs. In *Knowledge and Data Engineering*, ročník 16, 2004, ISSN 1041-4347, s. 1128–1142.

[4] van der Aalst, W. M. P.; Weijtersa, A. J. M. M.: Process mining: a research agenda. In *Computers in industry*, ročník 53, 2004, ISSN 0166-3615, s. 231–244.

[5] Agrawal, R.; Imielinski, T.; Swami, A.: Mining Association Rules Between Sets of Items in Large Databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data, Washington*, 1993, s. 207–216.

[6] Agrawal, R.; Imielinski, T.; Swami, A.: Mining associations between sets of items in massive databases. In *ACM SIGMOD Intl. Conf. on Management of Data*, 1994, s. 207–216.

[7] Antonie, M. L.; Zaiane, O.: Text Document Categorization by Term Association. In *IEEE International Conference on Data Mining (ICDM'02)*, Maebashi City, Japan, 2002, s. 19–26.

[8] Bartík, V.; Pospíšil, M.: Use of Frequent Itemset Mining to Analyze Business Processes. In *IT4Innovations - Center of excellence*, 2011.

[9] Edelstein, H. A.: *Introduction to data mining and knowledge discovery*. Two Crows Corporation, 2005, ISBN 1-892095-02-5.

[10] Grigori, D.; Casati, F.; Dayal, U.; aj.: Improving Business Process Quality through Exception Understanding, Prediction, and Prevention. In *Proceedings of the 27th VLDB Conference*, Roma, Italy, 2001.

[11] Han, J.; Pei, J.; Yin, Y.: Mining Frequent Patterns without Candidate Generation. In *Proceedings of the ACM-SIGMOD Conference on Management of Data (SIGMOD'00)*, Dallas, 2000, s. 1–12.

[12] Indarto, E.: Data Mining [online]. http://recommender-systems.readthedocs.org/en/latest/datamining.html, 2013 [cit. 2015-5-20].

[13] Kamber, J. H. M.: *Data Mining: Concepts and Techniques*. Elsevier Inc., 2006, ISBN 1-55860-901-6.

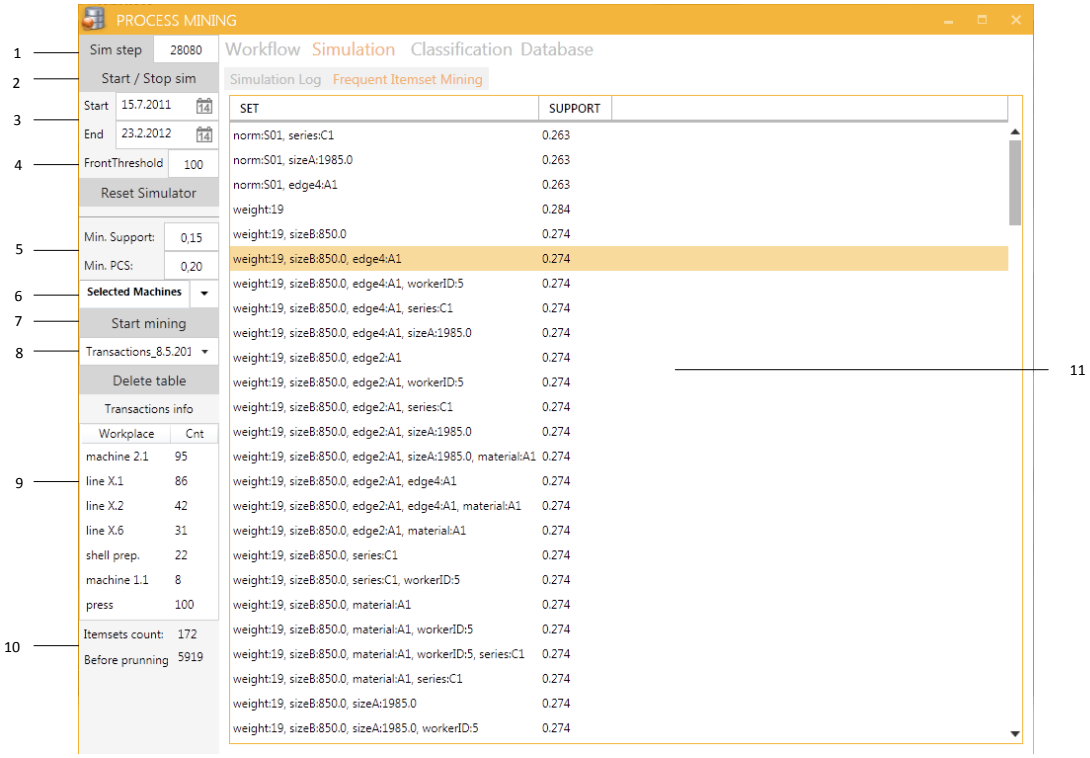
[14] Pospíšil, M.; Mates, V.; Hruška, T.; aj.: Process Mining in a Manufacturing Company for Predictions and Planning. In *International Journal on Advances in Software*, ročník 6, 2013, s. 283–297.

[15] Tiwari., A.; Turner, C.: A review of business process mining: state-of-the-art and future trends. In *Business Process Management Journal*, ročník 14, School of Applied Sciences, Cranfield University, Cranfield, UK, 2008, ISSN 1463-7154.

Příloha A

Manual

Popis ovládacích prvků u simulace a dolování frekventovaných množin

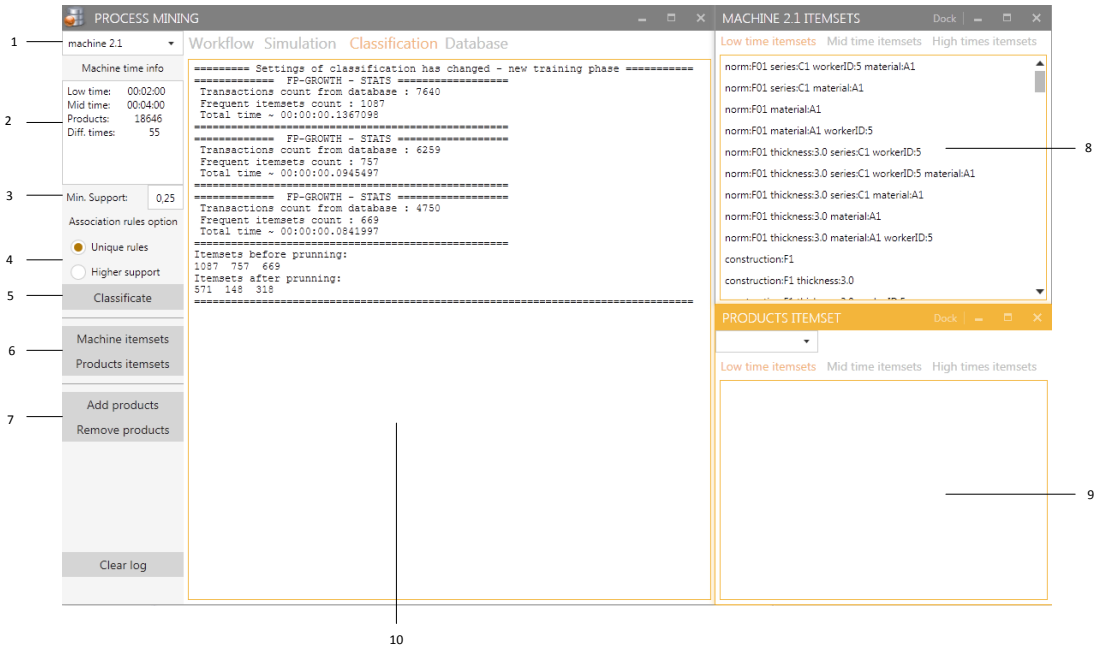


Obrázek A.1: Ukázka hlavního okna aplikace se simulátorem.

- 1 – Nastavení kroku simulátoru a tlačítko pro vykonání jednoho kroku.
- 2 – Tlačítko pro spuštění a zastavení simulátoru.
- 3 – Nastavení data začátku a konce simulátoru.
- 4 – Nastavení hodnoty nárustu fronty, při které se mají produkty ukládat do databáze.
- 5 – Nasavení parametrů dolování frekventovaných množin.

- 6 – Vybrání strojů, u kterých se má dolovat.
- 7 – Spuštění dolování.
- 8 – Výběr tabulky z databáze s daty pro dolování.
- 9 – Informační okno o pracovištích a počtu produktů u každého pracoviště.
- 10 – Statistiky počtu frekventovaných množin.
- 11 – Výsledky dolování.

Popis ovládacích prvků u klasifikace na základě asociačních pravidel



Obrázek A.2: Ukázka hlavního okna aplikace s klasifikátorem.

- 1 – Výběr pracoviště, u kterého chceme klasifikovat.
- 2 – Informační okno o intervalech zpracování.
- 3 – Nastavení minimální podpory.
- 4 – Nastavení filtrování asociačních pravidel.
- 5 – Tlačítko pro spuštění klasifikace.
- 6 – Tlačítka pro zobrazení oken s výslednými asociačními pravidly.
- 7 – Tlačítka pro přidání a odebraní produktů.
- 8 – Okno s asociačními pravidly u vybraného pracoviště.

- 9 – Okno s produkty a jejich nalezenými asociačními pravidly. Výsledky se zobrazí až po klasifikaci.
- 10 – Informace o klasifikaci.